PSEDIT

```
< NLS, PSEDIT.NLS.39, >, 6-Feb-78 11:52 JDH ;;;;
FILE psedit % L10 <rel-nls>psedit %% (l10,) (rel-nls,psedit.rel,) %    02
% Declarations %                                                      03
   REGISTER r1=1, r2=2, r3=3, r4=4;                                   04
   REF msgda, rawchr, inpt, tda;                                      05
   DECLARE                                                            06
      notile = 0, noct = 0,ctcsp = 1, ctfrz = 2,                      07
      ctcpfz = 3, ctmkr = 8, ctcmk = 9, ctcfm = 11, ctlcfm = 15;      08
                                                                      09
   DECLARE EXTERNAL                                                03799
      copyflag = 1, moveflag = 2, trnsflag = 3, deltflag = 4;      03800
% EDITOR SUBSYSTEM %                                                 010
   %append%                                                          062
      (xappend) %Execute Append Command%                           04919
         PROCEDURE                                                 04920
            %FORMALS%                                              04921
               (result, %result record%                            04922
               parsemode,        %parsing, backup, cleanup%        04923
               sourcentity,      %source entity type%              04924
               source,  %source pointer%                           04925
               destentity,       %destination entity type%         04926
               destination,      %destination pointer%             04927
               literal);         %string to insert between destination
               and source%                                         04928
               REF                                                 04929
                  result, sourcentity, source, destentity, destination,
                  literal;                                         04930
            LOCAL adstr[40];                                       04931
         %----------------------%                                  04932
         CASE parsemode OF                                         04933
            = parsing:                                             04934
               BEGIN                                               04935
               % No CML for this: NOTE percents changed to double
               percent:                                            04936
               CASE sourcentity OF                                 04937
                  = 8 %%- link -%%:                                04938
                     BEGIN                                         04939
                     IF source.stastr THEN                         04940
                        BEGIN                                      04941
                        IF NOT FIND                                04942
                           SF(source) $(SP/TAB) ("(/"</"--") THEN  04943
                              ST source _ "<, SF(source) SE(source);
                                                                   04944
                        IF NOT FIND                                04945
                           SE(source) $(SP/TAB) (")/">) THEN       04946
                              ST source _ SF(source) SE(source), ">;
                                                                   04947
                        END;                                       04948
                     lnkprs( &source, $adstr);                     04949
                     source _ adstr[ls];                           04950
                        source[1] _ adstr[ls+1];                   04951
                     [&source+d2sel] _ adstr[le];                  04952
                        [&source+d2sel+1] _ adstr[le+1];           04953
                     END;                                          04954
                  ENDCASE;                                         04955
               %                                                   04956
```

```
            CASE sourcentity OF                                          04957
               = 4 %- statement -%:                                      04958
                  BEGIN                                                  04959
                  clist(ctlcfm, destination.stfile, source.stfile);
                                                                         04960
                  dpset(dsprfst, destination, source, endfil);           04961
                  FIND SE(destination) ^curmkr;                          04962
                  IF NOT source.stastr THEN                              04963
                     cappsta(destination, source, &literal,
                     &literal+d2sel)                                     04964
                  ELSE                                                   04965
                     capptex(destination, &source, &source+d2sel,
                     &literal, &literal+d2sel);                          04966
                  clupdt();                                              04967
                  END;                                                   04968
               ENDCASE err(notyet);                                      04969
            END;                                                         04970
         ENDCASE;                                                        04971
      RETURN(&result);                                                   04972
      END.
                                                                         04973
%archive%                                                                 094
   (xarchive) %Execute Archive Command%                                   095
      PROCEDURE                                                           096
         %FORMALS%                                                        097
            (result, %result record%                                      098
            parsemode,        %parsing, backup, cleanup%                  099
            filename,         %name of file to be archived%              0100
            parameters);      %do/dont delete, deferred/immediate, not
            allowed%                                                     0101
            REF                                                          0102
               result, filename, parameters;                             0103
      LOCAL rhostn, arcparms, i;                                        03578
      LOCAL STRING filstr[200];                                         03579
      %----------------------%                                          0106
      CASE parsemode OF                                                 0107
         = parsing:                                                     0108
            BEGIN                                                       0109
            % parse file name %                                         0110
               rhostn _ lnbfls( &filename, 0, $filstr);                03581
            % parse the parameters %                                   03839
               arcparms _  i _ FALSE;                                   03840
               arcparms.flarfl.fdbarc _ TRUE;                           03841
               WHILE [parameters] := [parameters] -1 DO                 03842
                  CASE [parameters + (i_i+1)] OF                        03843
                     = 57 %- delete -%: arcparms.flarfl.fdbadl _
                     FALSE;                                             03844
                     = 58 %- deferred -%: arcparms.flarfl.fdbarc _
                     TRUE;                                              03845
                     %-= 59  immediate : err(notyet);-%                 03846
                     = 60 %- not -%:                                    03847
                        BEGIN                                           03853
                        arcparms.flarfl.fdbarc _ FALSE;                 03851
                        arcparms.flarfl.fdbnar _ TRUE;                  03852
                        END;                                            03854
                     = 61 %- prevent -%: arcparms.flarfl.fdbadl _
```

```
                    TRUE;                                                03848
                    = 62 %- reset -%: arcparms _ FALSE;                  03849
                    ENDCASE err(notyet);                                 03850
            *lit* _ NULL;                                                03838
            carcfil(rhostn, $filstr, arcparms, $lit);                     0114
            IF lit.L THEN                                                03833
               *lit* _ "The archive status of the following files has
               been changed:", CR, LF, *lit*                             03834
            ELSE                                                         03835
               *lit* _ "No files' archive status changed";               03836
            fbctl( typecalit, $lit);                                     03837
            END;                                                          0115
         ENDCASE;                                                         0116
      RETURN(&result);                                                    0117
      END.

                                                                          0118
%break%                                                                   0119
   (xbreak) %Execute Break Command%                                       0120
      PROCEDURE                                                           0121
         %FORMALS%                                                        0122
            (result, %result record%                                      0123
            parsemode,        %parsing, backup, cleanup%                  0124
            entity,  %entity type%                                        0125
            destination,      %destination pointer%                       0126
            level);  %level adjustment string%                            0127
            REF                                                           0128
               result, entity, destination, level;                        0129
         LOCAL TEXT POINTER t1, t2;                                       0130
         LOCAL STRING locstr[5];                                          0131
      %----------------%                                                  0132
      CASE parsemode OF                                                   0133
         = parsing:                                                       0134
            CASE entity OF                                                0135
               = 4 %- statement -%:                                       0136
                  BEGIN                                                   0137
                  clist(ctcmk, destination.stfile, endfil);               0138
                  dpset(dsprfst, destination, endfil,
                  dpstp(destination));                                    0139
                  %ignore lit for now%                                    0140
                     FIND SF(*locstr*) ^t1 ^t2;                           0141
                  curmkr _ cbresta(&destination, level, $t1, $t2);
                                                                          0142
                  curmkr[1] _ 1; % to start of broken stmt %              0143
                  clupdt();                                               0144
                  END;                                                    0145
               ENDCASE err(notyet);                                       0146
         ENDCASE;                                                         0147
      RETURN(&result);                                                    0148
      END.

                                                                          0149
%copy%                                                                    0250
   (xcopy) %Execute Copy Command%                                         0251
      PROCEDURE                                                           0252
         %FORMALS%                                                        0253
            (result, %result record%                                      0254
            parsemode,        %parsing mode (parsing, backup,
```

```
        cleanup)%                                                       0255
        sourcentity,     %source entity type%                           0256
        source,  %source pointer%                                       0257
        destentity,      %destination entity type%                      0258
        destination,     %destination pointer%                          0259
        level,   %level adjustment string%                             0260
        filterflag,      %if TRUE, filtered with viewspecs in vs%
                                                                        0261
        vs);             %viewspec string%                              0262
    LOCAL                                                               0263
        tlngth,                                                        03718
        rhostn, rhost2, adstr[40],                                     03550
        % stuff for copy directory %                                    0264
          info, % record saying what was requested %                    0265
          gropk,         % record saying how to group things %
                                                                        0266
          sortk % record saying how to wort things %                    0267
        ;                                                               0268
    LOCAL TEXT POINTER f1, f2;                                         03644
    LOCAL STRING                                                        0285
        filstr[200], filst2[200];                                       0286
    REF                                                                 0287
        result, sourcentity, source, destentity, destination,
        level, filterflag, vs;                                          0288
    %----------------------%                                            0289
    CASE parsemode OF                                                   0290
      = parsing:                                                        0291
        BEGIN                                                          03551
        result _ 0;                                                    03721
        CASE sourcentity OF                                            03553
          = 8 %- link -%:                                              03554
            BEGIN                                                      03555
            IF source.stastr THEN                                      03556
              BEGIN                                                    03557
              IF NOT FIND                                              03558
                SF(source) $(SP/TAB) ("(/"</"--") THEN                 03559
                  ST source _ "<, SF(source) SE(source);
                                                                       03560
              IF NOT FIND                                              03561
                SF(source) $(SP/TAB) (")/">) THEN                      03562
                  ST source _ SF(source) SE(source), ">;
                                                                       03563
              END;                                                     03564
            lnkprs( &source, $adstr);                                  03565
            source _ adstr[ls];                                        03566
              source[1] _ adstr[ls+1];                                 03567
            [&source+d2sel] _ adstr[le];                               03568
              [&source+d2sel+1] _ adstr[le+1];                         03569
            END;                                                       03570
          ENDCASE;                                                     03571
        CASE sourcentity OF                                             0292
          %text/structure entities%                                     0293
            = 5 %- character -%, = 7 %- invisible -%, = 12 %-
            text -%:                                                    0294
              BEGIN                                                     0295
              clist (ctcmk, destination.stfile,
```

```
                source.stfile);                                    0296
                dpset(dsprfmt, destination, endfil,
                destination);                                      0297
                curmkr _ destination;                              0298
                curmkr[1] _ destination[d2sel+1] +
                source[d2sel+1] - source[1] - 1;                   0299
                ccoptex(&destination+d2sel, &source,
                &source+d2sel, FALSE);                             0300
                clupdt ();                                         0301
                END;                                               0302
            = 14 %- word -%, = 13 %- visible -%, = 11 %- number
            -%, = 8 %- link -%:                                    0303
                BEGIN                                              0304
                clist (ctcmk, destination.stfile,
                source.stfile);                                    0305
                dpset(dsprfmt, destination, endfil,
                destination);                                      0306
                curmkr _ destination;                              0307
                curmkr[1] _ destination[d2sel+1] +
                source[d2sel+1] - source[1];                       0308
                ccoptex(&destination+d2sel, &source,
                &source+d2sel, TRUE);                              0309
                clupdt ();                                         0310
                END;                                               0311
            = 4 %- statement -%:                                   0312
                BEGIN                                              0313
                curmkr _ xcmst(&destination, level, &source,
                copyflag, filterflag, &vs);                        0314
                curmkr[1] _ 1;                                     0315
                END;                                               0316
            = 2 %- group -%, = 3 %- plex -%, = 1 %- branch -%:
                                                                   0317
                BEGIN                                              0318
                curmkr _ xcmgrp(&destination, level, &source,
                copyflag, filterflag, &vs);                        0319
                curmkr[1] _ 1;                                     0320
                END;                                               0321
        = 15 %- file -%:                                           0322
            BEGIN                                                  0323
            % get and initialize message string %                 03732
                result _ getstring( 3000, $dsphlk);               03723
                *[result]* _ "Copied Files Are:", CR, LF;          03724
                tlngth _ [result].L;                               03725
            % parse source file name %                             0324
                rhostn _ lnbfls( &source, 0, $filstr);             03572
            % parse destination file name %                        0328
                rhost2 _ lnbfls( &destination, 0, $filst2);
                                                                   03573
            ccopfil(rhostn, $filstr, rhost2, $filst2, result);
                                                                    0332
            % tell the user what we did %                         03733
                IF ( [result].L > tlngth ) THEN                    03728
                    fbctl( typecalit, result)                      03729
                ELSE fbctl( typecalit, $"No Files Copied");
                                                                   03730
            END;                                                    0333
```

```
                = 9 %- directory -%:                                    0334
                   BEGIN                                                0335
                   dpset(dspstrc,destination,endfil,endfil);            0336
                   info _ gropk _ sortk _ 0;                            0337
                   *filstr* _ "*.*;*";                                 03604
                   xdiropt( &source, destentity, $info, $gropk,
                   $sortk, $rhostn, $filstr);                           0338
                   curmkr _ ccopdir(&destination, level, info,          0339
                      gropk, sortk, rhostn, $filstr);                   0340
                   curmkr[1] _ 1;                                       0341
                   END;                                                 0342
                = 63 %- archive -%:                                     0343
                   BEGIN                                                0344
                   curmkr _ ccoparcdir(&destination, level, &source,
                   &source+d2sel, destentity);                          0345
                   curmkr[1] _ 1;                                       0346
                   END;                                                 0347
                = 64 %- sequential -%:                                  0348
                   BEGIN                                                0349
                   dpset(dsprfst, destination, endfil,
                   dpstp(destination));                                 0350
                   % move file name to local string %                  0351
                      CASE lnbfls( &source, 0, $filstr) OF             03574
                         = lhostn: NULL;                               03575
                         ENDCASE                                       03576
                            err($"Remote File Manipulations Not
                            Implemented Yet");                         03577
                   % setup text pointers to start and end of string %
                                                                        0360
                      FIND SF(*filstr*) ^f1 SE(*filstr*) ^f2;           0361
                   CASE destentity OF                                  04485
                      = 55 %- two -%: destentity _ heurfil;            04486
                      = 66 %- justified -%: destentity _ justfil;
                                                                        04487
                      = 67 %- assembler -%: destentity _ assfil;
                                                                        04488
                      = 0: %normal% destentity _ tenfil;               04489
                      ENDCASE err(notyet);                             04490
                   curmkr _ ccopseqfil (destination, level, $f1, $f2,
                   destentity);                                         0362
                   curmkr[1] _ 1;                                       0363
                   END;                                                 0364
                ENDCASE err(notyet);                                    0365
             END;                                                      03552
          = backup, = cleanup:                                         03719
             IF result THEN freestring(result, $dspblk);               03720
          ENDCASE;                                                      0366
       RETURN(&result);                                                 0367
       END.

                                                                        0368
                                                                        0369
   %copy/move support routines%
      (xdiropt)      % parse input for directory commands %            04722
                                                                        04723
         PROCEDURE
            (source, % record ptr to text ptrs for dir. name%          04724
            dent,    % record ptr to parameter list %                  04725
            info,    % adr: record to get request info %               04726
```

```
    gropk,    % adr: record to get request info %
    sortk,    % adr: record to get request info %
    rhostn,   % adr: cell to get host number %
    deffil    % adr: default file string %
    );
LOCAL
    pptr,     % temp pointer %
    count     % temp counter %
    ;
LOCAL TEXT POINTER tptr;
LOCAL STRING tstrng[40];
REF source, pptr, info, gropk, sortk, dent, rhostn, deffil;


% set up file group string %
    rhostn _ lhostn;
    IF source THEN
        BEGIN
        tptr _ [&source+d2sel];
        tptr[1] _ [&source+d2sel+1];
        IF FIND tptr < '<ESC> ^tptr THEN
            *deffil* _ '<, source tptr, '<^F>, *deffil*
        ELSE
            IF FIND tptr < '<^F> THEN
                *deffil* _ '<, source tptr, *deffil*
            ELSE
                *deffil* _ '<, source tptr, '>, *deffil*;
        END;
% parse the parameter input %
    count _ dent/4;  % each item is 4 elements long %
    &pptr _ &dent + 1;
    WHILE count DO
        BEGIN
        CASE pptr OF
            = 68 %- both -%: info.dlidlt _ 2;
            = 57 %- delete -%: info.dlidlt _ 1;
            = 69 %- undelete -%: info.dlidlt _ 0;
            = 70 %- for -%:
                BEGIN
                FIND SF(*[[&pptr+1]]*) ^tptr;
                rhostn _ lnbfls( $tptr, 0, &deffil);
                IF NOT FIND SF(*deffil*) '< THEN
                    IF source THEN
                        BEGIN
                        tptr _ [&source+d2sel];
                        tptr[1] _ [&source+d2sel+1];
                        IF FIND tptr < '<ESC> ^tptr THEN
                            *deffil* _
                                '<, source tptr, '<^F>, *deffil*
                        ELSE
                            IF FIND tptr < '<^F> THEN
                                *deffil* _
                                    '<, source tptr, *deffil*
                            ELSE
                                *deffil* _
```

```
                                    '<, source tptr, '>, *deffil*;
                                                                    04780
                    END;                                            04781
                 END;                                               04782
              = 63 %- archive -%:                                   04783
                 CASE  [&pptr + 1] OF                               04784
                    = 71 %- status -%: info.dliars _ TRUE;          04785
                    = 72 %- tape -%: info.dliart _ TRUE;            04786
                    ENDCASE;                                        04787
              = 73 %- account -%: info.dliacc _ TRUE;               04788
              = 74 %- no -%:                                        04789
                 CASE [&pptr + 1] OF                                04790
                    = 75 %- versions -%: info.dlinvr _ 1;           04791
                    = 76 %- extension -%: info.dlinex _ 1;          04792
                    ENDCASE;                                        04793
              = 77 %- date -%:                                      04794
                 CASE [&pptr + 1] OF                                04795
                    = 63 %- archive -%: info.dlitar _ 1;            04796
                    = 78 %- creation -%: info.dlitcr _ 1;           04797
                    = 79 %- last -%: info.dlitdm _ 1;               04798
                    = 80 %- first -%: info.dlitov _ 1;              04799
                    = 81 %- read -%: info.dlitrd _ 1;               04800
                    = 82 %- write -%: info.dlitwr _ 1;              04801
                    ENDCASE;                                        04802
              = 83 %- dump -%: info.dlidmt _ TRUE;                  04803
              = 84 %- everything -%:                                04804
                 BEGIN                                              04805
                 info.dliacc _ TRUE;                                04806
                 info.dliars _ TRUE;                                04807
                 info.dliart _ TRUE;                                04808
                 info.dlidmt _ TRUE;                                04809
                 info.dlidfr _ TRUE;                                04810
                 info.dlilwr _ TRUE;                                04811
                 info.dlibyt _ TRUE;                                04812
                 info.dlimis _ TRUE;                                04813
                 info.dlinrw _ TRUE;                                04814
                 info.dliprt _ TRUE;                                04815
                 info.dlisiz _ TRUE;                                04816
                 info.dlitar _ 2;                                   04817
                 info.dlitcr _ 2;                                   04818
                 info.dlitdm _ 2;                                   04819
                 info.dlitov _ 2;                                   04820
                 info.dlitrd _ 2;                                   04821
                 info.dlitwr _ 2;                                   04822
                 END;                                               04823
              = 79 %- last -%: info.dlilwr _ TRUE;                  04824
              = 85 %- length -%: info.dlibyt _ TRUE;                04825
              = 86 %- miscellaneous -%: info.dlimis _ TRUE;         04826
              = 11 %- number -%:                                    04827
                 CASE [&pptr + 1] OF                                04828
                    = 75 %- versions -%: info.dlidfr _ TRUE;
                                                                    04829
                    = 87 %- accesses -%: info.dlinrw _ TRUE;
                                                                    04830
                    ENDCASE;                                        04831
              = 88 %- protect -%: info.dliprt _ TRUE;               04832
```

```
                = 89 %- size -%: info.dlisiz _ TRUE;                    04833
                = 90 %- time -%:                                        04834
                   CASE [&pptr + 1] OF                                  04835
                      = 63 %- archive -%: info.dlitar _ 2;              04836
                      = 78 %- creation -%: info.dlitcr _ 2;             04837
                      = 79 %- last -%: info.dlitdm _ 2;                 04838
                      = 80 %- first -%: info.dlitov _ 2;                04839
                      = 81 %- read -%: info.dlitrd _ 2;                 04840
                      = 82 %- write -%: info.dlitwr _ 2;                04841
                      ENDCASE;                                          04842
                = 91 %- verbose -%:                                     04843
                   BEGIN                                                04844
                   info.dlisiz _ TRUE;                                  04845
                   info.dlilwr _ TRUE;                                  04846
                   IF NOT info.dlitwr THEN info.dlitwr _ 1;             04847
                   IF NOT info.dlitrd THEN info.dlitrd _ 1;             04848
                   END;                                                 04849
                = 2 %- group -%:                                        04850
                   BEGIN                                                04851
                   gropk _ 0;                                           04852
                   IF [&pptr + 1] = 203 %- reverse -%  THEN
                   gropk.dlgrvr _ TRUE;                                 04853
                   CASE [&pptr + 2] OF                                  04854
                      = 73 %- account -%: gropk.dlgacc _ TRUE;
                                                                        04855
                      = 63 %- archive -%:                               04856
                         CASE [&pptr + 3] OF                            04857
                            = 77 %- date -%: gropk.dlgdar _ TRUE;
                                                                        04858
                            = 71 %- status -%: gropk.dlgars _ TRUE;
                                                                        04859
                            = 72 %- tape -%: gropk.dlgart _ TRUE;
                                                                        04860
                            ENDCASE;                                    04861
                      = 78 %- creation -%: gropk.dlgdcr _ TRUE;
                                                                        04862
                      = 57 %- delete -%: gropk.dlgdlt _ TRUE;           04863
                      = 83 %- dump -%:                                  04864
                         CASE [&pptr + 3] OF                            04865
                            = 77 %- date -%: gropk.dlgddm _ TRUE;
                                                                        04866
                            = 72 %- tape -%: gropk.dlgdmt _ TRUE;
                                                                        04867
                            ENDCASE;                                    04868
                      = 80 %- first -%: gropk.dlgdov _ TRUE;            04869
                      = 79 %- last -%: gropk.dlglwr _ TRUE;             04870
                      = 11 %- number -%: gropk.dlgdfr _ TRUE;           04871
                      = 88 %- protect -%: gropk.dlgprt _ TRUE;
                                                                        04872
                      = 81 %- read -%: gropk.dlgdrd _ TRUE;             04873
                      = 82 %- write -%: gropk.dlgdwr _ TRUE;            04874
                      ENDCASE;                                          04875
                   END;                                                 04876
                = 92 %- sort -%:                                        04877
                   BEGIN                                                04878
                   sortk _ 0;                                           04879
```

```
                    IF [&pptr + 1] = 203 %+ reverse +% THEN
                    sortk.dlsrvr _ TRUE;                                    04880
                    CASE [&pptr + 2] OF                                     04881
                       = 73 %- account -%: sortk.dlsacc _ TRUE;
                                                                            04882
                       = 63 %- archive -%:                                  04883
                          CASE [&pptr + 3] OF                               04884
                             = 90 %- time -%: sortk.dlstar _ TRUE;
                                                                            04885
                             = 72 %- tape -%: sortk.dlsart _ TRUE;
                                                                            04886
                             ENDCASE;                                       04887
                       = 93 %- bytesize -%: sortk.dlsbyt _ TRUE;
                                                                            04888
                       = 78 %- creation -%: sortk.dlstcr _ TRUE;
                                                                            04889
                       = 57 %- delete -%: sortk.dlsdlt _ TRUE;              04890
                       = 83 %- dump -%:                                     04891
                          CASE [&pptr + 3] OF                               04892
                             = 90 %- time -%: sortk.dlstdm _ TRUE;
                                                                            04893
                             = 72 %- tape -%: sortk.dlsdmt _ TRUE;
                                                                            04894
                             ENDCASE;                                       04895
                       = 79 %- last -%: sortk.dlslwr _ TRUE;                04896
                       = 85 %- length -%: sortk.dlslen _ TRUE;              04897
                       = 11 %- number -%:                                   04898
                          CASE [&pptr + 3] OF                               04899
                             = 87 %- accesses -%: sortk.dlsnac _
                             TRUE;                                          04900
                             = 81 %- read -%: sortk.dlsnrd _ TRUE;
                                                                            04901
                             = 82 %- write -%: sortk.dlsnwr _ TRUE;
                                                                            04902
                             = 75 %- versions -%: sortk.dlsdfr _
                             TRUE;                                          04903
                             ENDCASE;                                       04904
                       = 80 %- first -%: sortk.dlstov _ TRUE;               04905
                       = 81 %- read -%: sortk.dlstrd _ TRUE;                04906
                       = 89 %- size -%: sortk.dlssiz _ TRUE;                04907
                       = 82 %- write -%: sortk.dlstwr _ TRUE;               04908
                       ENDCASE;                                             04909
                    END;                                                    04910
                 ENDCASE;                                                   04911
              BUMP DOWN count;                                              04912
              &pptr _ &pptr + 4;                                            04913
              END;                                                          04914
           % done so return %                                               04915
             RETURN;                                                        04916
                                                                            04917
           END.                                                             04918

        (xcmst) PROCEDURE(destination, level, source, type, filterflag,
        vs):                                                                0563
           %copy or move statement%                                         0564
           LOCAL clistcalled, newsid, proc;                                 0565
```

```
    LOCAL STRING vsstr[50], levstr[50];                          0566
    REF proc, destination, source, vs;                           0567
    &proc _ (IF type = copyflag THEN $ccopsta ELSE $cmovsta);
                                                                 0568
    IF clistcalled _ ( type NOT= copyflag AND NOT source.stastr
    AND source.stfile NOT= destination.stfile ) THEN             0569
       clist (15, source.stfile, nofile);                       03767
    IF NOT source.stastr THEN                                    0573
       BEGIN                                                    03735
       newsid _ proc(destination, level, source, filterflag,
       &vs);                                                     0574
       IF type = copyflag THEN                                  03738
          dpset(dspstrc, newsid, endfil, getnxt(newsid))        03740
       ELSE                                                     03741
          dpset(dspstrc, source, destination, endfil);          03742
       END                                                      03736
    ELSE %do an insert%                                          0575
       BEGIN                                                    03683
       newsid _ cinssta(destination, level, &source,
       &source+d2sel);                                           0576
       dpset(dspstrc, newsid, endfil, getnxt(newsid));          03681
       END;                                                     03682
    IF clistcalled THEN clupdt ();                               0577
    RETURN(newsid);                                              0578
    END.
                                                                 0579
(xcmgrp) PROCEDURE(destination, level, source, type,
filterflag, vs);                                                 0580
    %copy or move group%                                         0581
    LOCAL clistcalled, proc, newsid, stid;                       0582
    LOCAL STRING vsstr[50], levstr[50];                          0583
    REF proc, destination, source, vs;                           0584
    &proc _ (IF type = copyflag THEN $ccopgro ELSE $cmovgro);
                                                                 0585
    IF clistcalled _ ( type NOT= copyflag AND NOT source.stastr
    AND source.stfile NOT= destination.stfile) THEN              0586
       clist (15, source.stfile, nofile);                       03768
    IF NOT source.stastr THEN                                    0589
       BEGIN                                                    03744
       stid _ (IF type = copyflag THEN dpstp(destination) ELSE
       endfil);                                                  0587
       newsid _ proc(destination, level, source,
       [&source+d2sel], filterflag, &vs);                        0590
       dpset(dspstrc, newsid, IF type = copyflag THEN endfil
       ELSE source, stid);                                      03747
       END                                                      03746
    ELSE %do an insert%                                          0591
       BEGIN                                                    03686
       newsid _ cinssta(destination, level, &source,
       &source+d2sel);                                           0592
       dpset(dspstrc, newsid, endfil, getnxt(newsid));          03684
       END;                                                     03685
    IF clistcalled THEN clupdt ();                               0593
    RETURN(newsid);                                              0594
    END.
                                                                 0595
```

```
%create%                                                                0596
   (xcreate) %Execute Create Command%                                   0597
      PROCEDURE                                                         0598
         %FORMALS%                                                      0599
            (result, %result record%                                    0600
            parsemode,          %parsing, backup, cleanup%              0601
            filename);          %name of file to create%                0602
            REF                                                         0603
               result, filename;                                        0604
         LOCAL da, rhostn;  REF da;                                     0605
         LOCAL STRING filstr[200];                                      0608
      %----------------------%                                         0609
      CASE parsemode OF                                                 0610
         = parsing:                                                     0611
            BEGIN                                                       0612
            cspupdate _ &da _ lda();                                    0613
            % parse input file link %                                   0614
               rhostn _ lnbfls( &filename, 0, $filstr);                 0615
            curmkr _ ccrefil(rhostn, $filstr);                          0618
               %returns stid to origin%                                 0619
            curmkr[1] _ 1;                                              0620
            dpset( dspyes, curmkr, endfil, endfil );                    0621
            cspvs _ da.davspec;                                         0622
            cspvs[1] _ da.davspc2;                                      0623
            END;                                                        0624
         ENDCASE;                                                       0625
      RETURN(&result);                                                  0626
      END.                                                              0627
                                                                        0628
%delete%                                                                04974
   (xdelete) %Execute Delete Command%                                   04975
      PROCEDURE                                                         04976
         %FORMALS%                                                      04977
            (result, %result record%                                    04978
            parsemode,          %parsing, backup, cleanup%              04979
            entity,  %entity type%                                      04980
            destination,        %destination pointer%                   04981
            filterflag,         %if TRUE, filtered with viewspecs in vs%04982
            vs);                %viewspec string%                       04983
            REF                                                         04984
               result, entity, destination, filterflag, vs;
         LOCAL stid, type, da, deleteda, endda, cords, tlngth,          04985
         rhostn, adstr[40];                                             04986
         REF da, deleteda;                                              04987
         LOCAL TEXT POINTER z1, z2;                                     04988
         LOCAL STRING filstr[200], dafilstr[200];                       04989
      %----------------------%                                         04990
      CASE parsemode OF                                                 04991
         = parsing:                                                     04992
            BEGIN                                                       04993
            result _ 0;                                                 04994
            CASE entity OF                                              04995
               = 9 %- link -%:                                          04996
                  BEGIN                                                 04997
                  lnkprs( &destination, $adstr);
```

```
            destination _ adstr[ls];                                    04998
               destination[1] _ adstr[ls+1];                            04999
            [&destination+d2sel] _ adstr[le];                           05000
               [&destination+d2sel+1] _ adstr[le+1];                    05001
            END;                                                        05002
         ENDCASE;                                                       05003
      CASE entity OF                                                    05004
         %text and structure entities%                                  05005
            = 5 %- character -%, = 12 %- text -%, = 7 %-
            invisible -%:                                               05006
               BEGIN                                                    05007
               clist (ctcmk, destination.stfile, nofile);               05008
               dpset(dsprfmt, destination, endfil,
               destination);                                            05009
               FIND destination ^curmkr;                                05010
               cdeltex(&destination, &destination+d2sel,
               FALSE);                                                  05011
               IF FIND curmkr > ENDCHR THEN FIND curmkr
               _curmkr;                                                 05012
               clupdt ();                                               05013
               END;                                                     05014
            = 14 %- word -%, = 13 %- visible -%, = 11 %- number
            -%, = 8 %- link -%:                                         05015
               BEGIN                                                    05016
               clist (ctcmk, destination.stfile, nofile);               05017
               dpset(dsprfmt, destination, endfil,
               destination);                                            05018
               z1 _ destination[d2sel];                                 05019
                  z1[1] _ destination[d2sel+1];                         05020
               FIND destination ^curmkr;                                05021
               IF NOT FIND z1 > SP THEN                                 05022
                  IF FIND destination < SP ^curmkr THEN NULL;
                                                                        05023
               cdeltex(&destination, &destination+d2sel, TRUE);
                                                                        05024
               IF FIND curmkr > ENDCHR THEN FIND curmkr
               _curmkr;                                                 05025
               clupdt ();                                               05026
               END;                                                     05027
            = 4 %- statement -%:                                        05028
               BEGIN                                                    05029
               clist (ctlcfm, destination.stfile, nofile);
                                                                        05030
               dpset(dspstrc, destination, endfil,
               dpstp(destination));                                     05031
               CASE curmkr _ getnxt(destination) OF                     05032
                  = destination,                                        05033
                  = endfil:                                             05034
                     curmkr _ getbck(destination);                      05035
                  ENDCASE;                                              05036
               curmkr[1] _ 1;                                           05037
               cdelsta(destination, filterflag, &vs);                   05038
               clupdt ();                                               05039
               END;                                                     05040
            = 2 %- group -%, = 3 %- plex -%, = 1 %- branch -%:
                                                                        05041
```

```
                BEGIN                                                   05042
                clist (ctlcfm, destination.stfile, nofile);             05043
                dpset(dspstrc, destination, endfil,
                dpstp(destination));                                    05044
                CASE curmkr _ getnxt(getend(destination[d2sel]))        05045
                OF                                                      05046
                   = endfil:                                            05047
                      curmkr _ getbck(destination);                     05048
                   ENDCASE;                                             05049
                curmkr[1] _ 1;
                cdelgro(destination, destination[d2sel],
                filterflag, &vs);                                       05050
                clupdt ();                                              05051
                END;                                                    05052
             = 15 %- file -%:                                           05053
                BEGIN                                                   05054
                cspupdate _ FALSE;                                      05743
                result _ getstring( 3000, $dspblk);                     05055
                *[result]* _ "Deleted Files Are:", CR, LF;              05056
                tlngth _ [result].L;                                    05057
                rhostn _ lnbfis( &destination, 0, $filstr);             05058
                cdelfil( rhostn, $filstr, result);                      05059
                IF ( [result].L > tlngth ) THEN                         05060
                   fbctl( typecalit, result)                            05061
                ELSE fbctl( typecalit, $"No Files Deleted");            05062
                IF FIND SF(*[result]*) "Deleted Files Are:" ^z2
                THEN                                                    05638
                   BEGIN                                                05639
                   endda _ $dpyarea + dacnt*dal;                        05640
                   WHILE (FIND z2 ['<] ^z1 _z1 ['>] ^z2 ) DO            05641
                      BEGIN                                             05642
                      *filstr* _ z1 z2;                                 05643
                      FOR &da _ $dpyarea UP dal UNTIL >= endda DO
                                                                        05644
                         IF da.daexis AND NOT da.daempty THEN           05645
                            BEGIN                                       05646
                            filnam(da.dacsp.stfile, $dafilstr);
                                                                        05647
                            IF *filstr* = *dafilstr* THEN               05648
                               BEGIN                                    05649
                               da.daempty _ TRUE;                       05650
                               close(da.dacsp.stfile);                  05651
                               dpset(dspallf, da.dacsp, endfil,
                               endfil);                                 05652
                               END;                                     05653
                            mkdelfrr(da.dalink, $filstr);               05654
                            END;                                        05655
                      END;                                              05656
                   END;                                                 05657
                END;                                                    05063
             %not implemented = 94 %%- archived -%%: %%file%%           05064
                cdelarcfil(&destination, &destination+d2sel);%
                                                                        05065
             = 22 %- marker -%:                                         05066
                cdelmar(&destination, &destination+d2sel,
```

```
    lcfile());                                                  05067
  = 95 %- all -%: %markers%                                     05068
    cdelallmar(lcfile()); %must know file%                      05069
  = 96 %- modifications -%: %to file%                           05070
    BEGIN                                                       05071
    stid _ orgstid;                                             05072
    stid.stfile _ lcfile();                                     05073
    clist (ctlcfm, stid.stfile, nofile);                        05074
    dpset(dspallf, stid, endfil, endfil);                       05075
    cdelmodfil(stid.stfile);                                    05076
    <TOEXEC, unlkclist> (); %check clist items%                 05077
    clupdt ();                                                  05078
    END;                                                        05079
  = 21 %- edge -%: %of window%                                  05080
    BEGIN                                                       05081
    &da _ dsparea(boundry(destination[1], FALSE :
    destination[1], type));                                     05082
    cords _ lccords();                                          05083
    dpset(dspallf, endfil, endfil, endfil);                     05084
    clearda(0);                                                 05085
    clrall(0, TRUE);                                            05086
    ON SIGNAL ELSE alldsp();                                    05087
    CASE type OF                                                05088
      = lbound: %left edge of window DA to be deleted%
                                                                05089
        BEGIN                                                   05090
        %da points to right window%                             05091
        IF da.daleft = taleft THEN                              05092
          err($"cannot delete margin edge");                    05093
        IF NOT da.dalneighbor THEN                              05094
          err($"This window does not have a left
          neighbor");                                           05095
        &deleteda _ dsparea(da.dalneighbor); %left
        window%                                                 05096
        IF cords.xcord > da.daleft THEN %keep windows
        to right of boundary%                                   05097
          BEGIN                                                 05098
          fixbnd(TRUE, da.daleft, deleteda.daleft,
          TRUE);                                                05099
          END                                                   05100
        ELSE %keep windows to left of boundary%                 05101
          BEGIN                                                 05102
          &deleteda _ &da := &deleteda;                         05103
          %now da is left and deleteda is right
          window%                                               05104
          fixbnd(TRUE, da.daright, deleteda.daright,
          TRUE);                                                05105
          END;                                                  05106
        END;                                                    05107
      = rbound: %right edge of window to be deleted%
                                                                05108
        BEGIN                                                   05109
        IF da.daright = taright THEN                            05110
          err($"cannot delete margin edge");                    05111
        IF NOT da.darneighbor THEN                              05112
          err($"This window does not have a right
```

```
      neighbor");                                          05113
    &deleteda _ dsparea(da.darneighbor); %right
    window%                                                05114
    IF cords.xcord <= da.daright THEN %keep one
    to left of boundary%                                   05115
      BEGIN                                                05116
      fixbnd(TRUE, da.daright, deleteda.daright,
      TRUE);                                               05117
      END                                                  05118
    ELSE %keep one to right of boundary%                   05119
      BEGIN                                                05120
      &deleteda _ &da := &deleteda;                        05121
      fixbnd(TRUE, da.daleft, deleteda.daleft,
      TRUE);                                               05122
      END;                                                 05123
    END;                                                   05124
= tbound: %top edge of window to be deleted%
                                                           05125
    BEGIN                                                  05126
    IF da.datop = tatop THEN                               05127
      err($"cannot delete margin edge");                   05128
    IF NOT da.datneighbor THEN                             05129
      err($"This window does not have a top
      neighbor");                                          05130
    &deleteda _ dsparea(da.datneighbor); %top
    window%                                                05131
    IF cords.ycord > da.datop THEN %keep window
    on bottom of edge%                                     05132
      BEGIN                                                05133
      fixbnd(FALSE, da.datop, deleteda.datop,
      TRUE);                                               05134
      END                                                  05135
    ELSE %keep window on top of edge%                      05136
      BEGIN                                                05137
      &deleteda _ &da := &deleteda;                        05138
      fixbnd(FALSE, da.dabottom,
      deleteda.dabottom, TRUE);                            05139
      END;                                                 05140
    END;                                                   05141
= bbound: %bottom edge of window to be deleted%
                                                           05142
    BEGIN                                                  05143
    IF da.dabottom = tabottom THEN                         05144
      err($"cannot delete margin edge");                   05145
    IF NOT da.dabneighbor THEN                             05146
      err($"This window does not have a bottom
      neighbor");                                          05147
    &deleteda _ dsparea(da.dabneighbor); %bottom
    window%                                                05148
    IF cords.ycord <= da.dabottom THEN %keep
    window on top of edge%                                 05149
      BEGIN                                                05150
      fixbnd(FALSE, da.dabottom,
      deleteda.dabottom, TRUE);                            05151
      END                                                  05152
    ELSE %keep window on bottom of edge%                   05153
```

```
                        BEGIN                                          05154
                        &deleteda _ &da := &deleteda;                  05155
                        fixbnd(FALSE, da.datop, deleteda.datop,
                        TRUE);                                         05156
                        END;                                           05157
                     END;                                              05158
                  ENDCASE;                                             05159
               fixbuf();                                               05160
               END;                                                    05161
            ENDCASE err(notyet);                                       05162
         END;                                                          05163
      = backup, = cleanup:                                             05164
         IF result THEN freestring(result, $dspblk);                   05165
      ENDCASE;                                                         05166
   RETURN(&result);                                                    05167
   END.

                                                                       05168
%edit%                                                                  0745
   (xedit) %Execute Edit Command%                                       0746
      PROCEDURE                                                         0747
         %FORMALS%                                                      0748
            (result,          %result record%                           0749
            parsemode,                %parsing, backup, cleanup%        0750
            destination);     %destination pointer%                     0751
            REF                                                         0752
               result, destination;                                     0753
         LOCAL TEXT POINTER t1, t2;                                     0754
      %----------------------%                                         0755
      CASE parsemode OF                                                 0756
         = parsing:                                                     0757
            BEGIN                                                       0758
            dpset(dsprfmt, destination, endfil, endfil);                0764
            t1 _ destination;  t1[1] _ destination[1];                 03782
            editx($t1); %fill the global string LIT with the new
            version%                                                    0759
            FIND SF(*lit*) ^t1 SE(*lit*) ^t2;                           0760
            creptex(&destination, &destination+d2sel, $t1, $t2);        0761
            curmkr _ destination;                                       0762
            curmkr[1] _ 1;                                              0763
            END;                                                        0765
         ENDCASE;                                                       0766
      RETURN(&result);                                                  0767
      END.

                                                                        0768
%expunge%                                                               0769
   (xexpunge) %Execute Expunge Command%                                 0770
      PROCEDURE                                                         0771
         %FORMALS%                                                      0772
            (result, %result record%                                    0773
            parsemode,        %parsing, backup, cleanup%                0774
            entity); %type of directory%                                0775
            REF                                                         0776
               result, entity;                                          0777
      %----------------------%                                         0778
      CASE parsemode OF                                                 0779
         = parsing:                                                     0780
```

```
            CASE entity OF                                                  0781
               = 9 %- directory -%:                                         0782
                  % no parameters for the time being %                      0783
                  cexpdir(0);  %expunge connected directory%                0784
               = 63 %- archive -%:                                          0785
                  cexparcdir();  %expunge archive directory%                0786
               ENDCASE err(notyet);                                         0787
         ENDCASE;                                                           0788
      RETURN(&result);                                                      0789
      END.                                                                  0790

%force%                                                                     03887
   (xforce) %Force case Set Command%                                        03888
      PROCEDURE                                                             03889
         %FORMALS%                                                          03890
            (result, %result record%                                        03891
            parsemode,        %parsing, backup, cleanup%                    03892
            param1,  %parameter one%                                        03894
            param2,  %parameter two%                                        03895
            destination); %destination pointer%                             03897
            REF                                                             03898
               result, param1, param2, destination;                         03899
         LOCAL csize, hinc, vinc, da, end1, save, tp2, stid,
         adstr[40];                                                         03900
         LOCAL STRING sizestring[10];                                       03901
         REF da, tp2;                                                       03902
      %----------------------%                                             03903
      CASE parsemode OF                                                     03904
         = parsing:                                                         03905
            BEGIN                                                           03906
            dpset(dspno, endfil, endfil, endfil);                           03907
            param2 _ CASE param2 OF                                         03911
               =0: xsmode; % none specified %                               03912
               =80 %- first -%: iupcase;                                    03913
               =97 %- upper -%: upcase;                                     03914
               =98 %- lower -%: lowcase;                                    03915
               ENDCASE err( notyet );                                       03916
            CASE param1 OF                                                  03917
               = 8 %- link -%:                                              03918
                  BEGIN                                                     03919
                  lnkprs( &destination, $adstr);                            03920
                  destination _ adstr[1s];                                  03921
                     destination[1] _ adstr[1s+1];                          03922
                  [&destination+d2sel] _ adstr[le];                         03923
                     [&destination+d2sel+1] _ adstr[le+1];                  03924
                  END;                                                      03925
               ENDCASE;                                                     03926
            CASE param1 OF                                                  03927
               = 5 %- character -%, = 14 %- word -%, = 13 %- visible
               -%, = 7 %- invisible -%, = 8 %- link -%, = 11 %-
               number -%, = 12 %- text -%:                                  03928
                  BEGIN                                                     03929
                  clist (ctcmk, destination.stfile, nofile);                03930
                  dpset(dsprfmt, destination, endfil, destination);
                                                                            03931
                  curmkr _ destination;  curmkr[1] _
```

```
                destination[d2sel+1]-1;                                 03932
                csetctex(&destination, &destination+d2sel, param2);
                                                                        03933
                clupdt ();                                              03934
                END;                                                    03935
             = 4 %- statement -%:                                       03936
                BEGIN                                                   03937
                clist (ctcfm, destination.stfile, nofile);              03938
                dpset(dsprfmt, destination, endfil, destination);
                                                                        03939
                curmkr _ destination;  curmkr[1] _ 1;                   03940
                csetcsta(destination, param2);                          03941
                clupdt ();                                              03942
                END;                                                    03943
             = 2 %- group -%, = 3 %- plex -%, = 1 %- branch -%:
                                                                        03944
                BEGIN                                                   03945
                clist (ctcfm, destination.stfile, nofile);              03946
                dpset(dspallf, destination, endfil, endfil);            03947
                curmkr _ destination;  curmkr[1] _ 1;                   03948
                csetcgro(destination, [&destination+d2sel],
                param2);                                                03949
                clupdt ();                                              03950
                END;                                                    03951
             = 99 %- mode -%:                                           03952
                csetcmod(param2);                                       03953
             ENDCASE err(notyet);                                       03954
          END;                                                          04067
       ENDCASE;                                                         04068
    RETURN(&result);                                                    04069
    END.

                                                                        04070
%insert%                                                                 0815
  (xinsert)    %Execute Insert Command%                                  0816
    PROCEDURE                                                           0817
      %FORMALS%                                                         0818
        (result, %result record%                                        0819
        parsemode,       %parsing, backup, cleanup%                     0820
        entity,  %entity type%                                          0821
        destination,     %destination pointer%                          0822
        level,   %level adjustment characters%                         0823
        parameter);      %viewspec characters%                          0824
        REF                                                             0825
          result, entity, destination, level, parameter;                0826
    LOCAL                                                               03175
      temp, type, da, cords, x, y, adstr[40];                           03176
    REF da;                                                             04215
    LOCAL STRING                                                         0827
      locstr[500];          % string for date and time %                0828
    LOCAL TEXT POINTER                                                   0829
      tp1, tp2;                                                          0830
    %----------------------%                                            0831
    CASE parsemode OF                                                    0832
      = parsing:                                                         0833
        BEGIN                                                           03487
        CASE entity OF                                                  03489
```

```
        = 8 %- link -%:                                                 03490
          BEGIN                                                         03491
          IF parameter.stastr THEN                                      03492
            BEGIN                                                       03493
            IF NOT FIND                                                 03494
              SF(parameter) $(SP/TAB) ("(/"</"--") THEN                 03495
                ST parameter _                                          03496
                  "<, SF(parameter) SE(parameter);                      03508
            IF NOT FIND                                                 03497
              SE(parameter) $(SP/TAB) (")/">) THEN                      03498
                ST parameter _                                          03499
                  SF(parameter) SE(parameter), ">;                      03509
            END;                                                        03500
          lnkprs( &parameter, $adstr);                                  03501
          parameter _ adstr[ls];                                        03502
            parameter[1] _ adstr[ls+1];                                 03503
          [&parameter+d2sel] _ adstr[le];                               03504
            [&parameter+d2sel+1] _ adstr[le+1];                         03505
          END;                                                          03506
        ENDCASE;                                                        03507
      CASE entity OF                                                     0834
        %text/structure entities%                                        0835
          = 5 %- character -%, = 12 %- text -%, = 7 %-
          invisible -%:                                                  0836
            BEGIN                                                        0837
            clist (ctcmk, destination.stfile, nofile);                   0838
            dpset(dsprfmt, destination, endfil,
            destination);                                                0839
            curmkr _ destination[d2sel];                                 0840
              curmkr[1] _ destination[d2sel+1] +
              parameter[d2sel+1]-parameter[1] - 1;                       0841
            cinstex(&destination+d2sel, &parameter,
            &parameter+d2sel, FALSE);                                    0842
            clupdt ();                                                   0843
            END;                                                         0844
          = 14 %- word -%, = 13 %- visible -%, = 11 %- number
          -%, = 8 %- link -%:                                            0845
            BEGIN                                                        0846
            clist (ctcmk, destination.stfile, nofile);                   0847
            dpset(dsprfmt, destination, endfil,
            destination);                                                0848
            curmkr _ destination[d2sel];                                 0849
              curmkr[1] _ destination[d2sel+1] +
              parameter[d2sel+1]-parameter[1];                           0850
            cinstex(&destination+d2sel, &parameter,
            &parameter+d2sel, TRUE);                                     0851
            clupdt ();                                                   0852
            END;                                                         0853
          = 4 %- statement -%:                                           0854
            BEGIN                                                        0855
            temp _ 0;                                                   03178
            curmkr _ xcmst( &destination, level, &parameter,
            copyflag, 0, $temp);                                        03179
            curmkr[1] _ 1;                                               0859
            END;                                                         0861
```

```
        = 1 %- branch -%, = 3 %- plex -%, = 2 %- group -%:
                                                                0862
            BEGIN                                              03180
            temp _ 0;                                          03181
            curmkr _ xcmgrp( &destination, level,
            &parameter, copyflag, 0, $temp);                   03182
            curmkr[1] _ 1;                                     03183
            END;                                               03184
        = 77 %- date -%, = 90 %- time -%:                       0868
          BEGIN                                                 0869
          % date (and time) to string; set up pointers %        0870
            *locstr* _ NULL;                                    0871
            getdat( $locstr );                                  0872
            CASE entity OF                                      0873
               =77 %- date -%:                                  0874
                  BEGIN                                         0875
                  IF NOT                                        0876
                     (FIND SF(*locstr*) $PT (SP ^tp1))          0877
                        THEN err($"Bad Date From TENEX");
                                                                0878
                  ST tp1 _ SF(tp1) tp1;                         0879
                  END;                                          0880
               ENDCASE;                                         0881
            FIND SF(*locstr*) ^tp1 SE(*locstr*) ^tp2;           0882
          clist (ctcmk, destination.stfile, nofile);            0883
          dpset(dsprfmt, destination, endfil, destination);
                                                                0884
          curmkr _ destination[d2sel];                          0885
            curmkr[1] _ destination[d2sel+1];                   0886
          cinstex(&destination+d2sel, $tp1, $tp2, TRUE);        0887
          clupdt ();                                            0888
          END;                                                  0889
        = 100 %- sendmail -%: %form%                            0890
          BEGIN                                                03281
          *locstr* _                                           03285
            *sjtitle*, EOL, *sjcomment*, EOL,                  03680
            *sjauthor*, *initsr*, EOL, *sjnumber*, EOL,
                                                               04293
            *sjaction*, EOL, *sjinfo*, EOL, *sjsubcol*, EOL,
                                                               04294
            *sjkeywords*, EOL, *sjhandling*, EOL,              04296
            *sjrecording*, EOL, *sjhardcopy*, EOL,             04297
            *sjrfc*, EOL, *sjobsoletes*, EOL,                  04298
            *sjaccess*, EOL, *sjupdates*, EOL,                 04302
            *sjlink*, EOL, *sjforward*, EOL,                   04299
            *sjmessage*, EOL, *sjbranch*, EOL,                 04300
            *sjplex*, EOL, *sjgroup*, EOL,                     04303
            *sjfile*, EOL, *sjsendit* ;                        04304
          FIND SF(*locstr*) ^tp1 SE(*locstr*) ^tp2;            03287
          curmkr _ cinssta(destination, level, $tp1, $tp2);
                                                               03286
          curmkr[1] _ 1;                                       03679
          dpset(dspstrc, curmkr, endfil, curmkr);              03687
          END;                                                 03288
        = 21 %- edge -%: %of window%                           04157
          BEGIN                                                04158
```

```
                &da _ destination;                                      04167
                IF da.dafrozen THEN                                     04168
                   err($"Cannot split a frozen window");                04169
                boundry(destination[1], TRUE : cords, type);            04190
                   %ignore da returned%                                 04191
                clearda(&da); %erase display image from da%             04174
                clrall(&da, TRUE); %deallocate any strings%             04175
                IF parameter = 207 %+ center +%  THEN                   04186
                   BEGIN                                                04198
                   CASE type OF                                         04192
                      = tbound, = bbound:                               04193
                         BEGIN                                          04187
                         x _ (da.daright-da.daleft)/2;                  04170
                         cords.xcord _ (x/da.dahinc)*da.dahinc +
                         da.daleft;                                     04171
                         END;                                           04196
                      = lbound, = rbound:                               04194
                         BEGIN                                          04197
                         y _ (da.dabottom-da.datop)/2;                  04172
                         cords.ycord _ (y/da.davinc)*da.davinc +
                         da.datop;                                      04173
                         END;                                           04188
                      ENDCASE;                                          04195
                   END                                                  04199
                ELSE                                                    04200
                   BEGIN                                                04203
                   cords.xcord _
                   ((cords.xcord)/da.dahinc)*da.dahinc;                 04201
                   cords.ycord _
                   ((cords.ycord)/da.davinc)*da.davinc;                 04202
                   END;                                                 04204
                CASE type OF                                            04176
                   = lbound, = rbound:                                  04180
                      IF NOT hsplit(&da, cords, lccords()) THEN
                                                                        04178
                         err($"Display area too small");                04179
                   = tbound, = bbound:                                  04177
                      IF NOT vsplit(&da, cords, lccords()) THEN
                                                                        04181
                         err($"Display area too small");                04182
                      ENDCASE err(notyet);                              04183
                dpset(dspallf, endfil, endfil, endfil);                 04184
                END;                                                    04165
             ENDCASE err(notyet);                                        0892
          END;                                                          03488
       ENDCASE;                                                          0893
    RETURN(&result);                                                     0894
    END.                                                                 0895

(xinsstatement) %Execute repeat Insert Statement%                        0896
   PROCEDURE                                                             0897
      %FORMALS%                                                          0898
         (result, %result record%                                        0899
         parsemode,       %parsing, backup, cleanup%                     0900
         level,   %level adjustment value%                              0901
         source); %source text for stmt%                                 0902
```

```
            REF                                                          0903
               result, level, source;                                    0904
         %---------------%                                              0905
         CASE parsemode OF                                              0906
            = parsing:                                                  0907
               BEGIN                                                    0908
               curmkr _ cinssta(curmkr, level, &source, &source+d2sel);
                                                                        0911
               curmkr[1] _ 1;                                           0912
               dpset(dspstrc, curmkr, endfil, getnxt(curmkr));          0910
               END;                                                     0914
            ENDCASE;                                                    0915
         RETURN( &result );                                             0916
         END.
                                                                        0917
%load%                                                                  01093
   (xload) %Execute Load Command%                                       05529
      PROCEDURE                                                         05530
         %FORMALS%                                                      05531
            (result, %result record%                                    05532
            parsemode,       %parsing, backup, cleanup%                 05533
            entity,  %type of load%                                     05534
            filename);       %name of filestr to be loaded%             05535
         LOCAL hostno, da, fileno, stid, pcap, tp;                      05536
         LOCAL STRING hnmono[10], %host name or number%                 05537
            filestr[200], locflnm[200];                                 05538
         LOCAL TEXT POINTER ptr, ptr2;                                  05539
         REF                                                            05540
            result, filename, entity, da, tp;                           05541
         %----------------------%                                       05542
         CASE parsemode OF                                              05543
            = parsing:                                                  05544
               BEGIN                                                    05545
               &da _ cspupdate _ lda();                                 05546
               &tp _ &filename+d2sel;                                   05547
               % move file name to local string %                       05548
                  CASE hostno _ lnbfls( &filename, 0, $filestr) OF      05549
                     = lhostn: NULL;  %local host%                      05550
                     ENDCASE %hopefully NLS host%                       05551
                        BEGIN                                           05552
                        dismes(1,$"Loading Remote File");               05553
                        IF NOT FIND SE(*filestr*) ['.]                  05554
                           THEN *filestr* _ *filestr*, ".NLS";          05555
                        IF FIND SF(*filestr*) ['>] ^ptr ['.] ^ptr2 _
                        ptr2                                            05556
                           THEN *locflnm* _ ptr ptr2, "-REM", ptr2
                           SE(*filestr*);                               05557
                        *hnmono* _ STRING(hostno);                      05558
                        fetchfile($hnmono, $locflnm, $filestr);         05559
                        *filestr* _ *locflnm*;                          05560
                        dismes(1,$"Edits do not appear in remote
                        files.");                                       05561
                        dismes(1,$"Stored locally as");                 05562
                        END;                                            05563
               CASE entity OF                                           05566
                  = 15 %- file -%:                                      05567
```

```
                BEGIN                                                   05568
                IF (fileno _ cloafil($filestr)) THEN                    05569
                   BEGIN                                                05570
                   curmkr _ orgstid;                                    05571
                   curmkr.stfile _ fileno;                              05572
                   curmkr[1] _ 1;                                       05573
                   END;                                                 05574
                END;                                                    05575
             = 101 %- busy -%: % file %
             % this command does not work for remote files.  If
             file has no pc, locks and creates a pc file in users
             login directory %                                          05576
                BEGIN                                                   05577
                IF NOT nwheelf THEN                                     05578
                   err($"This command only available to system
                   personnel");                                         05579
                IF NOT FIND SF(*filestr*) ['.] THEN                     05580
                   *filestr* _ *filestr*, ".NLS";                       05581
                IF (stid _ cloamodfil($filestr)) THEN                   05582
                   BEGIN                                                05583
                   curmkr _ stid;                                       05584
                   curmkr[1] _ 1;                                       05585
                   END;                                                 05586
                disablw(pcap);                                          05587
                END;                                                    05588
             ENDCASE err(notyet);                                       05589
          cspvs _ da.davspec;                                           05590
          cspvsr[1] _ da.davspc2;                                       05591
          dpset(dspyes, curmkr, endfil, endfil);                        05592
          END;                                                          05593
       ENDCASE;                                                         05594
    RETURN(&result);                                                    05595
    END.
                                                                        05596
%logout%                                                                01154
   (xlogout) %Execute Logout Command%                                   05464
      PROCEDURE                                                         05465
         %FORMALS%                                                      05466
            (result, %result record%                                    05467
            parsemode);      %parsing, backup, cleanup%                 05468
            REF                                                         05469
               result:                                                  05470
      %----------------------%                                          05471
      CASE parsemode OF                                                 05472
         = parsing:                                                     05473
            BEGIN                                                       05474
            IF nlmode = fulldisplay THEN shutdis();                     05476
            clogout();                                                  05475
            %clean up this end and logout%                              05477
               IF NOT SKIP llgout(-1) THEN                              05478
                  err($"Unable to logout");                             05479
            END;                                                        05480
         ENDCASE;                                                       05481
      RETURN(&result);                                                  05482
      END.
                                                                        05483
```

```
%mark%                                                                  01174
   (xmark) %Execute Mark Command%                                       01175
      PROCEDURE                                                         01176
         %FORMALS%                                                      01177
            (result, %result record%                                    01178
            parsemode,       %parsing, backup, cleanup%                 01179
            destination,     %char to be marked%                        01180
            mkrname);        %marker name%                              01181
            REF                                                         01182
               result, destination, mkrname;                            01183
      %----------------------%                                          01184
      CASE parsemode OF                                                 01185
         = parsing:                                                     01186
            BEGIN                                                       01187
            cmarcha(&destination, &mkrname, &mkrname+d2sel);            01188
            curmkr _ destination; curmkr[1] _ destination[1];           01189
            END;                                                        01190
         ENDCASE;                                                       01191
      RETURN(&result);                                                  01192
      END.                                                              01193

%merge%                                                                 01194
   % COMMENTED OUT TO SAVE SPACE IN SYSTEM -- percents doubled          05766
   (xmerge) %%Execute Merge Command%%                                   01195
      PROCEDURE                                                         01196
         %%FORMALS%%                                                    01197
            (result, %%result record%%                                  01198
            parsemode,       %%parsing, backup, cleanup%%               01199
            entity,  %%source entity type%%                             01200
            source,  %%source pointer%%                                 01201
            destination); %%destination pointer%%                       01202
            REF                                                         01203
               result, entity, source, destination;                     01204
      %%----------------------%%                                        01205
      CASE parsemode OF                                                 01206
         = parsing:                                                     01207
            CASE entity OF                                              01208
               = 2 %%- group -%%, = 3 %%- plex -%%:                     01209
                  BEGIN                                                 01210
                  cmergro(&destination, &destination+d2sel, &source,
                  &source+d2sel);                                       01211
                  curmkr _ gethed(destination);  curmkr[1] _ 1;         01212
                  dpset(dspstrc, destination, endfil, endfil);          01213
                  END;                                                  01214
               = 1 %%- branch -%%:                                      01215
                  BEGIN                                                 01216
                  IF (destination := getsub(destination)) =
                  destination OR (source := getsub(source)) = source
                  THEN err($"Illegal Merge");                           01217
                  destination[d2sel] _ getail(destination);             01218
                  source[d2sel] _ getail(source);                       01219
                  REPEAT CASE(2 %%+ group +%%);                         01220
                  END;                                                  01221
               ENDCASE err(notyet);                                     01222
         ENDCASE;                                                       01223
      RETURN(&result);                                                  01224
```

```
    END.

                                                                    01225
  %                                                                 05765
%move%                                                              01226
  (xmove) %Execute Move Command%                                    01227
    PROCEDURE                                                       01228
      %FORMALS%                                                     01229
        (result, %result record%                                    01230
        parsemode,      %parsing, backup, cleanup%                  01231
        sourcentity,    %source entity type%                        01232
        source,         %source pointer%                            01233
        destentity,     %destination entity type%                   01234
        destination,    %destination pointer%                       01235
        level,          %level adjustment string%                   01236
        filterflag,     %if TRUE, filtered with viewspecs in vs%
                                                                    01237
        vs);            %viewspec string%                           01238
        REF                                                         01239
          result, sourcentity, source, destentity, destination,
          level, filterflag, vs;                                    01240
      LOCAL                                                         01241
        type, sourceda, destda, da, x, y, r, rhostn, rhost2,
        tlngth, adstr[40];                                          03407
        REF sourceda, destda, da;                                   03734
      LOCAL STRING                                                  03408
        filstr[200], filst2[200];                                   03409
      %----------------------%                                      01245
      CASE parsemode OF                                             01246
        = parsing:                                                  01247
          BEGIN                                                     03412
          result _ 0;                                               03703
          CASE sourcentity OF                                       03414
            = 8 %- link -%:                                         03415
              BEGIN                                                 03416
              IF source.stastr THEN                                 03424
                BEGIN                                               03425
                IF NOT FIND                                         03426
                  SF(source) $(SP/TAB) ("(/'</"--") THEN            03428
                    ST source _ '<, SF(source) SE(source);
                                                                    03427
                IF NOT FIND                                         03429
                  SE(source) $(SP/TAB) (")/'>) THEN                 03430
                    ST source _ SF(source) SE(source), '>;
                                                                    03431
                END;                                                03432
              lnkprs( &source, $adstr);                             03417
              source _ adstr[ls];                                   03418
                source[1] _ adstr[ls+1];                            03419
              [&source+d2sel] _ adstr[le];                          03420
                [&source+d2sel+1] _ adstr[le+1];                    03421
              END;                                                  03422
            ENDCASE;                                                03423
          CASE sourcentity OF                                       01248
            %text/structure entities%                               01249
              = 5 %- character -%, = 7 %- invisible -%, = 12 %-
              text -%:                                              01250
```

```
                BEGIN                                                   01251
                clist (ctcmk, destination.stfile,
                source.stfile);                                         01252
                dpset(dsprfmt, destination, source, endfil);
                                                                        01253
                curmkr _ destination[d2sel];                            01254
                   curmkr[1] _ destination[d2sel+1] +
                   source[d2sel+1]-source[1]-1;                         01255
                IF NOT source.stastr THEN                               01256
                   cmovtex(&destination+d2sel, &source,
                   &source+d2sel, FALSE)                                01257
                ELSE                                                    01258
                   cinstex(&destination+d2sel, &source,
                   &source+d2sel, FALSE);                               01259
                clupdt ();                                              01260
                END;                                                    01261
             = 14 %- word -%, = 13 %- visible -%, = 11 %- number
             -%, = 8 %- link -%:                                        01262
                BEGIN                                                   01263
                clist (ctcmk, destination.stfile,
                source.stfile);                                         01264
                dpset(dsprfmt, destination, source, endfil);
                                                                        01265
                curmkr _ destination[d2sel];                            01266
                   curmkr[1] _ destination[d2sel+1] +
                   source[d2sel+1]-source[1];                           01267
                IF NOT source.stastr THEN                               01268
                   cmovtex(&destination+d2sel, &source,
                   &source+d2sel, TRUE)                                 01269
                ELSE                                                    01270
                   cinstex(&destination+d2sel, &source,
                   &source+d2sel, TRUE);                                01271
                clupdt ();                                              01272
                END;                                                    01273
             = 4 %- statement -%:                                       01274
                BEGIN                                                   01275
                curmkr _ xcmst(&destination, level, &source,
                moveflag, filterflag, &vs);                             01276
                curmkr[1] _ 1;                                          01277
                END;                                                    01278
             = 2 %- group -%, = 3 %- plex -%, = 1 %- branch -%:
                                                                        01279
                BEGIN                                                   01280
                curmkr _ xcmgrp(&destination, level, &source,
                moveflag, filterflag, &vs);                             01281
                curmkr[1] _ 1;                                          01282
                END;                                                    01283
          = 15 %- file -%:                                              01284
             BEGIN                                                      01285
             % get and initialize message string %                      03716
                result _ getstring( 3000, $dspblk);                     03707
                *[result]* _ "Moved Files Are:", CR, LF;                03708
                tlngth _ [result].L;                                    03709
             % parse source file name %                                 01286
                rhostn _ lnbfls( &source, 0, $filstr);                  03475
             % parse destination file name %                            01290
```

```
        rhost2 _ inbfls( &destination, 0, $filst2);
                                                                03476
    cmovfil(rhostn, $filstr, rhost2, $filst2, result);
                                                                01294
    % tell the user what we did %                               03717
      IF ( [result].L > tlngth ) THEN                           03712
        fbctl( typecalit, result)                               03713
      ELSE fbctl( typecalit, $"No Files Moved");                03714
    END;                                                        01295
= 21 %- edge -%:                                                01296
    BEGIN                                                       01297
    &sourceda _ dsparea(boundry(source[1], FALSE :
    source[1], type));                                          01300
      %get boundary nearest cursor%                             01299
    IF sourceda.dafrozen THEN                                   01301
      err($"Can't move boundary of a frozen window");
                                                                01302
    &destda _ destination;                                      01303
    IF destda.dafrozen THEN                                     01320
      err($"Can't move boundary of a frozen window");
                                                                01321
    IF destentity = 207 %+ center +% THEN                       04205
      BEGIN                                                     04206
      &da _ findda(destination[1]);                             04207
      CASE type OF                                              04208
        = lbound, = rbound:                                     04209
          destination[1].xcord _
          (da.daright-da.daleft)/2;                             04214
        = tbound, = bbound:                                     04210
          destination[1].ycord _
          (da.dabottom-da.datop)/2;                             04213
        ENDCASE;                                                04211
      END;                                                      04212
    x _ destination[1].xcord - destda.daleft;                   01304
    IF destination[1].xcord < (destda.daright -
    destda.dahinc/2) THEN                                       01305
      BEGIN                                                     01306
      DIV x / destda.dahinc, x, r;                              01307
      IF r >= (destda.dahinc/2) THEN BUMP x;                    01308
      x _ x * destda.dahinc;                                    01309
      destination[1].xcord _ x + destda.daleft;                 01310
      END;                                                      01311
    y _ destination[1].ycord - destda.datop;                    01312
    IF destination[1].ycord < (destda.dabottom -
    destda.davinc/2) THEN                                       01313
      BEGIN                                                     01314
      DIV y / destda.davinc, y, r;                              01315
      IF r >= (destda.davinc/2) THEN BUMP y;                    01316
      y _ y * destda.davinc;                                    01317
      destination[1].ycord _ y + destda.datop;                  01318
      END;                                                      01319
    clearda(0); %erase entire text area%                        01322
    clrall(0, TRUE); %erase all line seg ref tables%
                                                                01323
    movbndry(&sourceda, source[1], destination[1],
    type);                                                      01324
```

```
                dpset(dspallf, endfil, endfil, endfil);          01325
                END;                                             01326
             ENDCASE err(notyet);                                01327
          END;                                                   03413
       = backup, = cleanup:                                      03704
          IF result THEN freestring(result, $dspblk);            03705
       ENDCASE;                                                  01328
    RETURN(&result);                                             01329
    END.

                                                                 01330
%output%                                                         01331
   (xoutsapf) %set oqapfg for output commands%                   01332
      PROCEDURE                                                  01333
         %FORMALS%                                               01334
            (result, %result record%                             01335
            parsemode,        %parsing, backup, cleanup%         01336
            parameter);       %value for oqapfg%                 01339
         REF                                                     01344
            result, parameter;                                   01345
      %----------------------%                                   01346
      CASE parsemode OF                                          01347
         = parsing:                                              01348
            oqapfg _ parameter;                                  03269
         ENDCASE oqapfg _ FALSE;                                 01411
      RETURN(&result);                                           01412
      END.

                                                                 01413
   (xoutsnhf) %set oqnhfg for output commands%                   04326
      PROCEDURE                                                  04327
         %FORMALS%                                               04328
            (result, %result record%                             04329
            parsemode,        %parsing, backup, cleanup%         04330
            parameter);       %value for oqnhfg%                 04331
         REF                                                     04332
            result, parameter;                                   04333
      %----------------------%                                   04334
      CASE parsemode OF                                          04335
         = parsing:                                              04336
            oqnhfg _ parameter;                                  04337
         ENDCASE oqnhfg _ FALSE;                                 04338
      RETURN(&result);                                           04339
      END.

                                                                 04340
   (xout1) %Execute Output (quickprint, journal, printer, com)
   Command%                                                      03187
      PROCEDURE                                                  03188
         %FORMALS%                                               03189
            (result, %result record%                             03190
            parsemode,        %parsing, backup, cleanup%         03191
            entity,  %entity type%                               03192
            filename,         %file name pointer%                03193
            parameter,        %viewspec characters%              03194
            tstnar): %$test or NULL%                             04152
         LOCAL tp;                                               03195
         LOCAL STRING locstr[200];                               03198
         REF                                                     03199
```

```
        result, entity, filename, parameter, tp, tstpar;          03200
    %----------------------%                                       03201
    CASE parsemode OF                                             03202
      = parsing:                                                  03203
        BEGIN                                                     03204
        % set so ^O won't clear output buffer %                   05636
          rubnocob _ TRUE;                                        05637
        % decode number of copies string if necessary %          03205
          IF parameter > 1 THEN                                   03206
            BEGIN                                                 03207
            &tp _ &parameter + d2sel;                             03208
            *locstr* _  parameter tp;                             03209
            ndr( &parameter, &parameter, &parameter );            03855
            parameter _ VALUE($locstr);                           03210
            END;                                                  03211
        % get output file name to locstr, use dfltname to
        construct the name if none was specified %                03212
          IF  filename                                            03213
            THEN % use name supplied by user %                    03214
              BEGIN                                               05774
              IF FIND filename > ([."lpt:".] / [."LPT:".] ) THEN
              *locstr* _ "LPT:"                                   05776
              ELSE CASE lnbfls( &filename, 0, $locstr) OF
                                                                  03402
                = lhostn: NULL;                                   03404
                ENDCASE                                           03405
                  err($"Remote File Manipulations Not
                  Implemented Yet")                               03406
              END                                                 05775
            ELSE % construct default name %                       03225
              dfltname( $locstr, entity, tstpar );                03226
        CASE entity OF                                            03227
          = 102 %- quickprint -%:                                 03228
            BEGIN                                                 03229
            IF NOT FIND SF(*locstr*) [".] THEN                    03230
              *locstr* _ *locstr*, ".", STRING(parameter);
                                                                  03231
                %make ext be number of copies%                    03232
            coutqui($locstr, lda());                              03233
            END;                                                  03234
          = 103 %- journal -%: %submission form%                  03235
            BEGIN                                                 03236
            IF NOT FIND SF(*locstr*) [".] THEN                    03237
              *locstr* _ *locstr*, ".", STRING(parameter);
                                                                  03238
                %make ext be number of copies%                    03239
            coutjouqui($locstr, lda());                           03240
            END;                                                  03241
          = 104 %- printer -%:                                    03242
            BEGIN                                                 03243
            IF NOT FIND SF(*locstr*) [".] THEN                    03244
              *locstr* _ *locstr*, ".", STRING(parameter);
                                                                  03245
                %make ext be number of copies%                    03246
            coutproc($locstr, lda(), opprdv, 0);                  03247
            END;                                                  03248
```

```
            = 105 %- com -%:                %Computer output to
            Microfilm%                                              03249
               BEGIN                                                03250
               comexflag _ [comexflag.RH];                          05631
               IF NOT FIND SF(*locstr*) ['.] THEN                   03251
                  *locstr* _ *locstr*, '.', STRING(parameter);
                                                                    03252
                     %make ext be number of copies%                 03253
               coutproc($locstr, lda(),  IF tstpar = 204 %- test
               -%  THEN opxpdv ELSE opcmdv, 0);                     03254
               END;                                                 03255
            = 67 %- assembler -%:                                   03256
               BEGIN                                                03257
               coutassfil($locstr, lda(), parameter);               03258
               END;                                                 03259
            = 64 %- sequential -%:                                  03260
               BEGIN                                                03261
               coutseqfil($locstr, lda(), parameter);               03262
               END;                                                 03263
            ENDCASE err(notyet);                                    03264
         END;                                                       03265
      ENDCASE;                                                      03266
   RETURN(&result);                                                 03267
   END.
                                                                    03268
(xout2) %Output (Terminal, Remote printer) Command%                 01414
   PROCEDURE                                                        01415
      %FORMALS%                                                     01416
         (result, %result record%                                   01417
         parsemode,        %parsing, backup, cleanup%               01418
         entity,  %entity type%                                     01419
         tip,              %tip name or filename if an output
         terminal file%                                             01420
         tipport, %tip port%                                        01421
         formfeed,         %TRUE: send FF, FALSE: see simff%        01422
         simff,    %TRUE: simulate FF%                              01423
         waitpb); %TRUE: wait at page break%                        01424
      LOCAL opflags, devtype, tp;                                   01425
      LOCAL STRING tipstr[10], trmstr[10], outfile[30];             01426
      REF result, entity, tip, tipport, formfeed, simff, waitpb,
      tp;                                                           01427
   %----------------------%                                        01428
   CASE parsemode OF                                              01429
      = parsing:                                                    01430
         BEGIN                                                      01431
         % setup flags record (to be passed to UP) %               01432
         opflags _ 0;                                               01433
         CASE formfeed OF                                           01434
            = 1:                                                    01435
               BEGIN                                                01436
               opflags.opform _ TRUE;                               01437
               opflags.opsimff _ FALSE;                             01438
               END;                                                 01439
            = 2, = 0:                                               01440
               BEGIN                                                01441
               opflags.opform _ FALSE;                              01442
```

```
                CASE simff OF                                                   01443
                   = 1:    opflags.opsimff _ TRUE;                              01444
                   = 2, = 0:       opflags.opsimff _ FALSE;                     01445
                   ENDCASE err($"invalid response");                            01446
                END;                                                            01447
             ENDCASE err($"invalid response");                                  01448
          CASE waitpb OF                                                        01449
             = 1:  opflags.opwtpb _ TRUE;                                       01450
             = 2, = 0:     opflags.opwtpb _ FALSE;                              01451
             ENDCASE err($"invalid response");                                  01452
          % construct file name based of device %                               01453
          CASE entity OF                                                        01454
             = 106 %- terminal -%:                                              01455
                BEGIN                                                           01456
                devtype _ optydv;                                               01457
                IF NOT tip THEN *outfile* _ "TTY:"                              04639
                ELSE                                                            04640
                   CASE lnbfls( &tip, 0, $outfile) OF                           04641
                      = lhostn: NULL;                                           04642
                      ENDCASE                                                   04643
                         err($"Remote File Manipulations Not
                         Implemented Yet");                                     04644
                END;                                                            01459
             = 107 %- remote -%: %printer/terminal%                             01460
                BEGIN                                                           01461
                &tp _ &tip + d2sel;                                             01462
                *tipstr* _ tip tp;                                              01463
                &tp _ &tipport + d2sel;                                         01464
                *trmstr* _ tipport tp;                                          01465
                devtype _ oprmdv;                                               01466
                *outfile* _ "NET:0.",                                           01467
                   STRING(VALUE($tipstr), 8), "-,
                    STRING((VALUE($trmstr)* 65536 + 2), 8);                     01468
                END;                                                            01469
             ENDCASE err(notyet);                                               01470
          coutproc($outfile, lda(), devtype, opflags);                          01471
          END;                                                                  01472
       ENDCASE;                                                                 01473
    RETURN(&result);                                                            01474
    END.                                                                        01475

(dfltname) % construct default output file name %                               01476
    PROCEDURE(                                                                  01477
    % FORMAL ARGUMENTS %                                                        01478
       str,          % ptr to result astring %                                 01479
       type,    % $quickprint, $com, etc. %                                    04476
       tstpar); % $test OR NULL %                                               04481
    LOCAL TEXT POINTER tp1, tp2;                                                01480
    REF % VARIABLES %                                                           01481
       str;                                                                     01482
    % ------------------ %                                                      01483
    %put file name into a string%                                               01484
       *str* _ NULL;                                                            01485
       tlinam (flda()].dacsp.stfile, &str);                                     01486
    % check it %                                                                01487
       IF NOT (FIND SF(*str*) ["],] $SP ^tp1 ["."] < CH ^tp2) THEN
```

```
                                                                    01488
        err ($"bad file name");                                     01489
    % edit string, putting printer directory and user's initials
    into it %                                                       01490
        *str* _ '$, *initsr*, '$, tp1 tp2;                          01491
        IF type = 105 %+ com +% AND NOT tstpar THEN                 04478
            *str* _ "<COM>", *str*, ".COM"                          04479
        ELSE *str* _ '<, *prtdir*, '>, *str*;                       04480
    RETURN;                                                         01492
    END.
                                                                    01493
%playback%                                                          01494
  (xplayback) %Execute Playback Command%                            01495
    PROCEDURE                                                       01496
      %FORMALS%                                                     01497
        (result, %result record%                                    01498
        parsemode,      %parsing, backup, cleanup%                  01499
        filename,       %name of file to be played%                 01500
        symrt);         % flag to simulate recorded timing%
                                                                    04616
        REF                                                         01501
          result, filename, symrt;                                  01502
    LOCAL                                                           01503
      rhostn;                                                       01504
    LOCAL TEXT POINTER f1, f2;                                      03645
    LOCAL STRING                                                    01505
      filstr[200];                                                  01506
    %---------------------%                                        01507
    CASE parsemode OF                                               01508
      = parsing:                                                    01509
        BEGIN                                                       01510
        % move file name to local string %                          01511
          rhostn _ lnbfls( &filename, 0, $filstr);                  03401
        % setup text pointers to start and end of string %         01520
          FIND SF(*filstr*) ^f1 SE(*filstr*) ^f2;                   01521
        % set global flag symtflg %                                04617
          symtflg _ symrt;                                          04618
        xrecplasup( FALSE, $f1, $f2, rhostn);                       01522
        END;                                                        01523
      ENDCASE;                                                      01524
    RETURN(&result);                                                01525
    END.
                                                                    01526
%print%                                                             01527
  (xprint) %Execute TNLS Print Command%                             01528
    PROCEDURE                                                       01529
      %FORMALS%                                                     01530
        (result, %result record%                                    01531
        parsemode,      %parsing, backup, cleanup%                  01532
        entity,  %entity type%                                      01533
        destination,    %destination pointer%                       01534
        vs);            %viewspec record%                           01535
        REF                                                         01536
          result, entity, destination, vs;                          01537
      LOCAL da: REF da;                                             01538
      LOCAL vssav1, vssav2, cspsav;      %Save current VS and SP%
```

```
                                                                    04306
    %----------------------%                                        01539
    CASE parsemode OF                                               01540
       = parsing:                                                   01541
         BEGIN                                                      01542
         &da _ lda();                                               01543
         CASE entity OF                                             01544
            %structure entities%                                    01545
               = 4 %- statement -%:                                 01546
                  BEGIN                                             01547
                  curmkr _ destination;  curmkr[1] _
                  destination[1];                                   01548
                  cprista(destination, &vs, &da);                   01549
                  %cspvs _ da.davspec;  cspvs[1] _ da.davspc2; not
                  necessary %                                       03790
                  cspvs _ vs;                                       04716
                  cspvs[1] _ vs[1];                                 04717
                  cspupdate _ &da;                                  04714
                  END;                                              01550
               = 2 %- group -%, = 1 %- branch -%, = 3 %- plex -%:
                                                                    01551
                  BEGIN                                             01552
                  curmkr _ destination;  curmkr[1] _
                  destination[1];                                   01553
                  cprigro(destination, [&destination+d2sel], &vs,
                  &da);                                             01554
                  %cspvs _ da.davspec;  cspvs[1] _ da.davspc2; not
                  necessary %                                       03791
                  cspvs _ vs;                                       04718
                  cspvs[1] _ vs[1];                                 04719
                  cspupdate _ &da;                                  04715
                  END;                                              01555
            = 108 %- rest -%, = 15 %- file -%:                      01556
              BEGIN                                                 04307
              cspsav _ da.dacsp; %save current SP%                  04315
              vssav1 _ da.davspec;        %save current VS%         04308
              vssav2 _ da.davspc2;                                  04309
              %da.davspec _ stdvsp;%      %use standard VS%         04310
              %da.davspc2 _ stdvsp[1];%                             04311
              IF entity = 15 %+ file +% THEN da.dacsp.stpsid _
              orgstid;                                              04316
              ON SIGNAL ELSE                                        04317
                 BEGIN                                              04318
                 da.dacsp _ cspsav;                                 04319
                 da.davspec _ vssav1;                               04320
                 da.davspc2 _ vssav2;                               04321
                 RETURN(&result);                                   04322
                 END;                                               04323
              cprires(da.dacsp, &da);                               01557
              ON SIGNAL ELSE;                                       04324
              da.dacsp _ cspsav;                                    04325
              da.davspec _ vssav1;        %restore current VS%
                                                                    04312
              da.davspc2 _ vssav2;                                  04313
              END;                                                  04314
            = 103 %- journal -%:                                    03788
```

```
                cprijou(da.dacsp, &da);                                 03789
              ENDCASE err(notyet);                                      01558
            END;                                                        01559
          ENDCASE;                                                      01560
        RETURN(&result);                                                01561
        END.                                                            01562

%process%                                                               01694
    (xprocess) %Execute Process Command%                                01695
        PROCEDURE                                                       01696
          %FORMALS%                                                     01697
            (result, %result record%                                    01698
            parsemode,        %parsing, backup, cleanup%                01699
            destentity,       % destination entity type %              01701
            destination);% destination record %                         01702
            REF                                                         01703
              result, entity, destentity, destination;                  01704
          LOCAL TEXT POINTER endtptr; % points to last char %           01705
        %----------------------%                                        01706
        CASE parsemode OF                                               01707
          = parsing:                                                    01708
            BEGIN                                                       01711
            endtptr _ destination[d2sel];                               01712
            endtptr[1] _ destination[d2sel+1];                          01713
            CASE destentity OF                                          01714
              = 4 %- statement -%:                                      01715
                NULL;                                                   01716
              = 1 %- branch -%, = 2 %- group -%, = 3 %- plex -%:        01717
                BEGIN                                                   01718
                endtptr _ getend(endtptr);                              01719
                FIND SE(endtptr) ^endtptr;                              01720
                END;                                                    01721
            ENDCASE err(notyet);                                        01722
            auxstartup( &destination, $endtptr );                       01723
            END;                                                        01724
          ENDCASE;                                                      01726
        RETURN(&result);                                                01727
        END.                                                            01728

                                                                        04305
%record%                                                                01729
    (xstart) %Execute Start Record Command%                             05501
        PROCEDURE                                                       05502
          %FORMALS%                                                     05503
            (result, %result record%                                    05504
            parsemode,        %parsing, backup, cleanup%                05505
            filename,         %name of file to be archived%             05506
            runfil); % true if should generate runfil format%           05507
            REF                                                         05508
              result, filename, runfil;                                 05509
        LOCAL rhostn;                                                   05510
        LOCAL TEXT POINTER f1, f2;                                      05511
        LOCAL STRING                                                    05512
          filstr[200];                                                  05513
        %----------------------%                                        05514
```

```
      CASE parsemode OF                                                 05515
         = parsing:                                                     05516
            BEGIN                                                       05517
            % move file name to local string %                          05518
               rhostn _ lnbfls( &filename, 0, $filstr);                 05519
            % point to start and end of the string %                    05520
               FIND SF(*filstr*) ^f1 SE(*filstr*) ^f2;                  05521
            % set up for runfil format if requested %                   05522
               recrunfil _ runfil;                                      05523
            xrecplasup( TRUE, $f1, $f2, rhostn);                        05524
            END;                                                        05525
         ENDCASE;                                                       05526
      RETURN(&result);                                                  05527
      END.

                                                                        05528
%renumber%                                                              01794
   (xrenumber) %Execute Renumber Command%                               01795
      PROCEDURE                                                         01796
         %FORMALS%                                                      01797
            (result, %result record%                                    01798
            parsemode);       %parsing, backup, cleanup%                01799
            REF                                                         01800
               result;                                                  01801
      %----------------------%                                          01802
      CASE parsemode OF                                                 01803
         = parsing:                                                     01804
            BEGIN                                                       01805
            crensidfil(lcfile());                                       01806
            dpset(dspyes, [lda()].dacsp, endfil, endfil);               01807
            END;                                                        01808
         ENDCASE;                                                       01809
      RETURN(&result);                                                  01810
      END.

                                                                        01811
%replace%                                                               01812
   (xreplace) %Execute Replace Command%                                 04343
      PROCEDURE                                                         04344
         %FORMALS%                                                      04345
            (result, %result record%                                    04346
            parsemode,        %parsing, backup, cleanup%                04347
            destentity,       %destination entity type%                 04348
            destination,      %destination pointer%                     04349
            sourcentity,      %source entity type%                      04350
            source); %source pointer%                                   04351
            REF                                                         04352
               result, sourcentity, source, destentity, destination;
                                                                        04353
         LOCAL delt;                                                    04354
         LOCAL TEXT POINTER tp1, tp2;                                   04355
         LOCAL STRING temp[40];                                         04356
         LOCAL adstr[40];                                               04357
      %----------------------%                                          04358
      CASE parsemode OF                                                 04359
         = parsing:                                                     04360
            BEGIN                                                       04361
            CASE sourcentity OF                                         04362
```

```
            = 8 %- link -%:                                          04363
               BEGIN                                                 04364
               IF source.stastr THEN                                 04365
                  BEGIN                                              04366
                  IF NOT FIND                                        04367
                     SF(source) $(SP/TAB) ('(/'</"--") THEN          04368
                        ST source _ '<, SF(source) SE(source);
                                                                     04369
                  IF NOT FIND                                        04370
                     SE(source) $(SP/TAB) (')/'>) THEN               04371
                        ST source _ SF(source) SE(source), '>;
                                                                     04372
                  END;                                               04373
               lnkprs( &source, $adstr);                             04374
               source _ adstr[ls];                                   04375
                  source[1] _ adstr[ls+1];                           04376
               [&source+d2sel] _ adstr[le];                          04377
                  [&source+d2sel+1] _ adstr[le+1];                   04378
               END;                                                  04379
            ENDCASE;                                                 04380
         CASE destentity OF                                          04381
            = 8 %- link -%:                                          04382
               BEGIN                                                 04383
               IF destination.stastr THEN                            04384
                  BEGIN                                              04385
                  IF NOT FIND                                        04386
                     SF(destination) $(SP/TAB) ('(/'</"--") THEN
                                                                     04387
                        ST destination _                             04388
                           '<, SF(destination) SE(destination);
                                                                     04389
                  IF NOT FIND                                        04390
                     SE(destination) $(SP/TAB) (')/'>) THEN          04391
                        ST destination _                             04392
                           SF(destination) SE(destination), '>;
                                                                     04393
                  END;                                               04394
               lnkprs( &destination, $adstr);                        04395
               destination _ adstr[ls];                              04396
                  destination[1] _ adstr[ls+1];                      04397
               [&destination+d2sel] _ adstr[le];                     04398
                  [&destination+d2sel+1] _ adstr[le+1];              04399
               END;                                                  04400
            = 11 %- number -%:                                       04401
               BEGIN                                                 04402
               delt _ destination[d2sel+1] - destination[1] -
               source[d2sel+1] +source[1];                           04403
               CASE delt OF                                          04404
                  < 0:                                               04405
                     BEGIN                                           04406
                     LOOP                                            04407
                        IF (delt := delt+1) >= 0 OR                  04408
                        (NOT FIND destination < SP SP ^destination
                        _destination) THEN EXIT LOOP;                04409
                     END;                                            04410
                  > 0:                                               04411
```

```
                        BEGIN                                                   04412
                        tp1 _ source;                                           04413
                        tp1[1] _ source[1];                                     04414
                        tp2 _ source[d2sel];                                    04415
                        tp2[1] _ source[d2sel+1];                               04416
                        *temp* _ NULL;                                          04417
                        UNTIL (delt _ delt-1) < 0 DO *temp* _ *temp*,           04418
                        SP;                                                     04419
                        *temp* _ *temp*, tp1 tp2;                               04420
                        FIND SF(*temp*) ^tp1 SE(*temp*) ^tp2;                   04421
                        source _ tp1;                                           04422
                        source[1] _ tp1[1];                                     04423
                        source[d2sel] _ tp2;                                    04424
                        source[d2sel+1] _ tp2[1];                               04425
                        END;                                                    04426
                    ENDCASE;                                                    04427
                END;                                                            04428
            ENDCASE;                                                            04429
        CASE sourcentity OF                                                     04430
            %text/structure entities%
                = 5 %- character -%, = 14 %- word -%, = 13 %-
                visible -%, = 7 %- invisible -%, = 8 %- link -%, =
                11 %- number -%, = 12 %- text -%, = 4 %- statement
                -%:                                                             04431
                    BEGIN                                                       04432
                    clist (ctmkr, destination.stfile,
                    source.stfile);                                             04433
                    dpset(dsprfmt, destination, endfil, endfil);
                                                                                04434
                    curmkr _ destination;  curmkr[1] _
                    destination[1]+source[d2sel+1]-source[1]-1;
                                                                                04435
                    creptex(&destination, &destination+d2sel,
                    &source, &source+d2sel);                                    04436
                    clupdt ();                                                  04437
                    END;                                                        04438
                = 1 %- branch -%, = 3 %- plex -%, = 2 %- group -%:              04439
                    BEGIN                                                       04440
                    clist (ctlcfm, destination.stfile,
                    source.stfile);                                             04441
                    dpset(dsprfst, destination, endfil,
                    dpstp(destination));                                        04442
                    curmkr _ crepgro(NOT source.stastr,
                    &destination, &destination+d2sel, &source,
                    &source+d2sel);                                             04443
                    clupdt ();                                                  04444
                    curmkr[1] _ 1;                                              04445
                    END;                                                        04446
                ENDCASE err(notyet);                                            04447
            END;                                                                04448
        ENDCASE;                                                                04449
    RETURN(&result);                                                            04450
    END.
                                                                                04451
                                                                                01349
%reset%
```

```
(xreset) %Execute Reset Command%                                        01850
   PROCEDURE                                                            01851
      %FORMALS%                                                         01852
         (result, %result record%                                       01853
         parsemode,       %parsing, backup, cleanup%                   01854
         entity,  %entity type%                                         01855
         destentity,      %level adjustment characters%                 01856
         destination); %destination pointer%                            01857
         REF                                                            01858
            result, entity, destentity, destination ;                   01859
      LOCAL da, save, rhostn;  REF da;                                  01860
   LOCAL STRING filstring[200];                                         03361
   %---------------------%                                             01863
   CASE parsemode OF                                                    01864
      = parsing:                                                        01865
         BEGIN                                                          01866
         dpset(dspno, endfil, endfil, endfil);                          01867
         CASE entity OF                                                 01868
            = 63 %- archive -%: %request for file%                      01869
               BEGIN                                                    01870
               rhostn_lnbfls(&destination, 0, $filstring);              03362
               *lit* _ NULL;                                            04142
               carcfil( rhostn, $filstring, 0, $lit);                   04143
               IF lit.L THEN                                            04144
                  *lit* _ "The archive status of the following
                  files has been changed:", CR, LF, *lit*               04145
               ELSE                                                     04146
                  *lit*  _ "No files' archive status changed";
                                                                        04147
               fbctl( typecalit, $lit);                                 04148
               END;                                                     04149
            = 109 %- case -%: %mode%                                    01876
               crescasmod();                                            01877
            %  commented out; percents doubled                          05763
            = 5 %%- character -%%: %%size for window%%                  01878
               setcharsize(lda(), tacsize);                             01879
            %                                                           05764
            = 110 %- content -%: %Content Analysis%                     01882
               BEGIN                                                    01883
               &da _ lda();                                             01884
               da.dacacode _ 0;                                         01885
               da.davspec.vscapf _ FALSE;                               01886
               END;                                                     01887
            = 8 %- link -%: %default for file%                          01888
               creslindef(lcfile());                                    01889
            = 18 %- name -%: %delimiters in destentity%                 01890
               BEGIN                                                    01891
               CASE destentity OF                                       01892
                  = 4 %- statement -%:                                  01893
                     cresnsta(destination);                             01894
                  = 2 %- group -%, = 1 %- branch -%, = 3 %- plex
                  -%:                                                   01895
                     cresngro(destination, [&destination+d2sel],
                     lda());                                            01896
                  ENDCASE err(notyet);                                  01897
               END;                                                     01898
```

```
            = 111 %- temporary -%: %modifications to file%                01910
              crestemmod(lcfile(), FALSE);                                01911
            = 47 %- tty -%: %window%                                      01912
              BEGIN                                                       01913
              clearda(0);                                                 04491
              clrall(0, TRUE);                                            04492
              %delete current one%                                        03752
                 IF ttysim := 0 THEN dealocda(ttyda);                     03753
              %restore old text display%                                  01928
                 IF (&da _ ttyda) NOT= &msgda AND da.daexis THEN          03756
                                                                          03757
                    BEGIN                                                 03758
                    da.daseq _ FALSE;                                     03761
                    da.dasuppress _ FALSE;                                03759
                    da.dahandle _ alocda(&da);                            03760
                    END;                                                  01936
                 dpset(dspallf, endfil, endfil, endfil);                  01918
               %re-allocate system default tty-sim area%                  01919
                 save _ linkcnsl := 0;                                    01920
                 ttysim _ alocda(&msgda);                                 01921
                 linkcnsl _ save;                                         03754
                 ttyda _ &msgda;                                          03755
                 ttywindow(&msgda);                                       01937
               defttysim _ TRUE;                                          01938
               END;                                                       01939
            = 112 %- viewspecs -%:                                        01940
              BEGIN                                                       01941
              cspupdate _ &da _ lda();                                    01942
              cspvs _ stdvsp;                                             01943
              cspvs[1] _ stdvsp[1];                                       04495
              curmkr _ da.dacsp;  curmkr[1] _ da.daccnt;                  04494
              dpset(dspyes, da.dacsp, endfil, endfil);                    01944
              END;                                                        01945
            ENDCASE err(notyet);                                          01946
          END;                                                            01947
        ENDCASE;                                                          01948
      RETURN(&result);
      END.                                                                01949
                                                                          01950
%retrieve%                                                                01951
   (xretrieve) %Execute Retrieve Command%                                 01952
      PROCEDURE                                                           01953
        %FORMALS%                                                         01954
          (result, %result record%                                        01955
          parsemode,       %parsing, backup, cleanup%                   01956
          filename);       %filename pointer%                           01957
          REF                                                             01958
            result, filename;                                             03360
      LOCAL rhostn;                                                       03354
      LOCAL STRING filstring[200];                                        01961
      %---------------------%                                            01962
      CASE parsemode OF                                                   01963
        = parsing:                                                        01964
          BEGIN                                                           03355
          rhostn _ lnbfls( &filename, 0, $filstring);                     01968
          cretarcfil(rhostn, $filstring);
```

```
            END;                                                    01969
         ENDCASE;                                                   01970
      RETURN(&result);                                              01971
      END.
                                                                    01972
%set%                                                               01973
   (xset) %Execute Set Command%                                     05169
      PROCEDURE                                                     05170
         %FORMALS%                                                  05171
            (result, %result record%                                05172
            parsemode,       %parsing, backup, cleanup%             05173
            entity,  %entity type%                                  05174
            param1,  %parameter one%                                05175
            param2,  %parameter two%                                05176
            param3,  %parameter three%                              05177
            destination); %destination pointer%                     05178
            REF                                                     05179
               result, entity, param1, param2, param3, destination;
                                                                    05180
         LOCAL mask, prot, count, i, rhostn, csize, ninc, vinc, da,
         end1, save, tp2, stid, adstr[40];                          05181
         LOCAL STRING sizestring[10], filstr[200];                  05182
         REF da, tp2;                                               05183
         %----------------------%                                   05184
         CASE parsemode OF                                          05185
            = parsing:                                              05186
               BEGIN                                                05187
               dpset(dspno, endfil, endfil, endfil);                05188
               CASE entity OF                                       05189
                  % commented out; percents doubled                 05759
                  = 5 %%- character -%%: %%size for window%%        05190
                     BEGIN                                          05191
                     %% convert number string to value %%          05192
                        &tp2 _ &param1 + d2sel;                     05193
                        *sizestring* _ param1 tp2;                  05194
                        csize _ VALUE($sizestring);                 05195
                     setcharsize(lda(), csize); %% go fix up da %%  05196
                     END;                                           05197
                  %                                                 05760
                  = 113 %- external -%:  % names link file address %
                                                                    05198
                     BEGIN                                          05199
                     stid _ orgstid;                                05200
                     stid.stfile _ [lda()].dacsp.stfile;            05201
                     dpset(dsprfmt, stid, endfil, endfil);          05202
                     IF param1.stastr THEN                          05203
                        BEGIN                                       05204
                        IF NOT FIND                                 05205
                           SF(param1) $(SP/TAB) ('(/'</"--") THEN   05206
                              ST param1 _                           05207
                                 '<, SF(param1) SE(param1);         05208
                        IF NOT FIND                                 05209
                           SE(param1) $(SP/TAB) (')/'>) THEN        05210
                              ST param1 _                           05211
                                 SF(param1) SE(param1), '>;         05212
                        END;                                        05213
```

```
                lnkprs( &param1, $adstr);                                   05214
                csetextname( stid.stfile, $adstr );                         05215
                END;                                                        05216
             = 110 %- content -%: %Content Analysis%                        05217
                CASE param1 OF                                              05218
                   = 114 %- to -%: %pattern%                                05219
                      BEGIN                                                 05220
                      cspcacode _ cpconan(&param2, cspupdate _
                      lda());                                               05221
                      END;                                                  05222
                   = 1: %on%                                                05223
                      BEGIN                                                 05224
                      cspupdate _ lda();                                    05225
                      cspvs.vscapf _ TRUE;                                  05226
                      END;                                                  05227
                   = 2: %off%                                               05228
                      BEGIN                                                 05229
                      cspupdate _ lda();                                    05230
                      cspvs.vscapf _ FALSE;                                 05231
                      END;                                                  05232
                   ENDCASE err(notyet);                                     05233
             = 9 %- link -%: %default for file%                             05234
                csetlindef(lcfile(), &param2, &param2+d2sel);               05235
             = 18 %- name -%: %delimiters in destentity%                    05236
                BEGIN                                                       05237
                IF param2 THEN                                              05238
                   BEGIN                                                    05239
                   CCPOS param2;  param2 _ READC;                           05240
                   IF param2 = ENDCHR THEN param2 _ 0;                      05241
                   END;                                                     05242
                IF param3 THEN                                              05243
                   BEGIN                                                    05244
                   CCPOS param3;  param3 _ READC;                           05245
                   IF param3 = ENDCHR THEN param3 _ 0;                      05246
                   END;                                                     05247
                curmkr _ destination;  curmkr[1] _ 1;                       05248
                CASE param1 OF                                              05249
                   = 4 %- statement -%:                                     05250
                      csetnsta(destination, param2, param3);                05251
                   = 2 %- group -%, = 1 %- branch -%, = 3 %- plex
                   -%:                                                      05252
                      csetngro(destination, [&destination+d2sel],
                      param2, param3, lda());                               05253
                   ENDCASE err(notyet);                                     05254
                END;                                                        05255
             = 115 %- private -%: %file%                                    05256
                chprvsts (lcfile (), $psprivate);                           05257
             = 116 %- public -%: %file%                                     05258
                chprvsts (lcfile (), $pspublic);                            05259
             = 111 %- temporary -%: %modifications to file%                 05260
                csettemmod(lcfile());                                       05261
             = 117 %- tenex -%: %protection for a file%                     05262
                BEGIN                                                       05263
                rhostn _ lnbfls( &param1, 0, $filstr);                      05264
                % parse the input %                                         05265
                   CASE [param2 + 1] OF                                     05266
```

```
                    = 118 %- allow -%:                                  05267
                       BEGIN                                            05268
                       prot _ 0;                                        05269
                       IF NOT (count _ [param2] - 2) THEN               05270
                          err($"Illegal protection specified");
                                                                        05271
                       i _ 1;                                           05272
                       WHILE i <= count DO                              05273
                           prot _ prot .V                               05274
                             (CASE [param2+2+(i := i + 1)] OF
                                                                        05275
                                = 81 %- read -%: 40B;                   05276
                                = 82 %- write -%: 20B;                  05277
                                = 119 %- execute -%: 10B;               05278
                                = 120 %- append -%: 04B;                05279
                                = 121 %- list -%: 02B;                  05280
                                = 95 %- all -%: 77B;                    05281
                                = 122 %- set -%:                        05282
                                   VALUE([param2+2+(i:=i+1)],8);
                                                                        05283
                                ENDCASE 0 );                            05284
                       prot _ prot .A 77B;                              05285
                       CASE [param2 + 2] OF                             05286
                          = 123 %- self -%:                             05287
                            BEGIN                                       05288
                            prot _ prot * 10000B;                       05289
                            mask _ 770000B;                             05290
                            END;                                        05291
                          = 2 %- group -%:                              05292
                            BEGIN                                       05293
                            prot _ prot * 100B;                         05294
                            mask _ 007700B;                             05295
                            END;                                        05296
                          = 116 %- public -%:                           05297
                            BEGIN                                       05298
                            prot _ prot * 1B;                           05299
                            mask _ 000077B;                             05300
                            END;                                        05301
                          ENDCASE                                       05302
                            err($"Illegal protection
                            specified");                                05303
                       END;                                             05304
                    = 124 %- forbid -%:                                 05305
                       BEGIN                                            05306
                       prot _ 0;                                        05307
                       IF NOT (count _ [param2] - 2) THEN               05308
                          err($"Illegal protection specified");
                                                                        05309
                       i _ 1;                                           05310
                       WHILE i <= count DO                              05311
                           prot _ prot .V                               05312
                             (CASE [param2+2+(i := i+1)] OF             05313
                                = 81 %- read -%: 40B;                   05314
                                = 82 %- write -%: 20B;                  05315
                                = 119 %- execute -%: 10B;               05316
                                = 120 %- append -%: 04B;                05317
```

```
                        = 121 %- list -%: 02B;                  05318
                        = 95 %- all -%: 77B;                    05319
                        = 122 %- set -%:                        05320
                           VALUE([param2+2+(i:=i+1)],8);
                                                                05321
                        ENDCASE 0 );                            05322
                    prot _ prot .A 77B .X 77B;                  05323
                    CASE [param2 + 2] OF                        05324
                       = 123 %- self -%:                        05325
                         BEGIN                                  05326
                         prot _ prot * 10000B;                  05327
                         mask _ 770000B;                        05328
                         END;                                   05329
                       = 2 %- group -%:                         05330
                         BEGIN                                  05331
                         prot _ prot * 100B;                    05332
                         mask _ 007700B;                        05333
                         END;                                   05334
                       = 116 %- public -%:                      05335
                         BEGIN                                  05336
                         prot _ prot * 1B;                      05337
                         mask _ 000077B;                        05338
                         END;                                   05339
                       ENDCASE                                  05340
                         err($"Illegal protection
                         specified");                           05341
                     END;                                       05342
                  = 62 %- reset -%:                             05343
                     BEGIN                                      05344
                     prot _ 777752B;                            05345
                     mask _ 18M;                                05346
                     END;                                       05347
                  = 115 %- private -%:                          05348
                     BEGIN                                      05349
                     mask _ 18M;                                05350
                     prot _ CASE [param2 + 2] OF                05351
                        = 123 %- self -%: 770000B;              05352
                        = 2 %- group -%: 777700B;               05353
                        = 116 %- public -%: 777777B;            05354
                        ENDCASE 777752B;                        05355
                     END;                                       05356
                   = 122 %- set -%:                             05357
                     BEGIN                                      05358
                     mask _ 18M;                                05359
                     prot _ VALUE([param2 + 2],8);              05360
                     END;                                       05361
                   ENDCASE                                      05362
                     err($"Illegal Protection Specified");
                                                                05363
                *lit* _ NULL;                                   05364
                cprofil(rhostn, $filstr, mask, prot, $lit);     05365
                IF lit.L THEN                                   05366
                   *lit* _ "The protection of the following files
                   has been changed:", CR, LF, *lit*            05367
                ELSE                                            05368
                   *lit* _ "No files' protection changed";      05369
```

```
                fbctl( typecalit, $lit);                                    05370
                END;                                                        05371
            = 47 %- tty -%: %window%                                        05372
                BEGIN                                                       05373
                %restore any suppressed da's%                               05374
                   IF (&da _ ttyda) NOT= &msgda AND da.daexis THEN          05375
                                                                            05376
                      BEGIN                                                 05377
                      da.daseg _ da.dasuppress _ FALSE;                     05378
                      da.dahandle _ alocda(&da);                            05379
                      END;                                                  05380
                &da _ param1;                                               05381
                IF da.dahandle THEN                                         05382
                   BEGIN                                                    05383
                   clearda(&da);                                            05384
                   dealocda(&da);                                           05385
                   da.dahandle _ 0;                                         05386
                   END;                                                     05387
                %delete old tty-sim%                                        05388
                   IF ttysim := 0 THEN dealocda(ttyda);                     05389
                %allocate new tty-sim area%                                 05390
                   da.daseg _ da.dasuppress _ TRUE;
                   end1 _ da.dalsz :=
                   (da.dabottom-da.datop)/da.davinc;                        05391
                   save _ linkcnsl := 0;                                    05392
                   ttysim _ alocda(ttyda _ &da);                            05393
                   linkcnsl _ save;                                         05394
                   da.dalsz _ end1;                                         05395
                   ttywindow(&da);                                          05396
                dpset(dspyes, da.dacsp, endfil, endfil);                    05397
                defttysim _ FALSE;                                          05398
                END;                                                        05399
            = 112 %- viewspecs -%:                                          05613
                BEGIN                                                       05614
                cspupdate _ &da _ lda();                                    05615
                IF da.daempty THEN                                          05616
                   BEGIN %set VS words here and now%                        05617
                   cspupdate _ FALSE; %no cmdfinish action%                 05618
                   da.dapvs _ da.davspec := param1;                         05619
                   da.dapvs2 _ da.davspc2 :=param1[1];                      05620
                   da.dacacode _ cspcacode;                                 05621
                   da.dausqcod _ cspusqcod;                                 05622
                   END                                                      05623
                ELSE                                                        05624
                   BEGIN                                                    05625
                   cspvs _ param1;                                          05626
                   cspvs[1] _ param1[1];                                    05627
                   END;                                                     05628
                dpset(dspyes, [cspupdate].dacsp, endfil, endfil);           05629
                                                                            05630
                END;                                                        05407
            ENDCASE err(notyet);                                            05408
        END;                                                                05409
    ENDCASE;                                                                05410
RETURN(&result);
END.
```

```
                                                                        05411
;  commented out; percents doubled                                      05761
(setcharsize) %% sets all fields in the da for specifed char
size %%                                                                 05412
   PROCEDURE ( da, charsize);                                           05413
   LOCAL hinc, vinc;                                                    05414
   REF da;                                                              05415
   dpset(dspallf, da.dacsp, endfil, endfil);                            05416
   CASE charsize OF                                                     05417
      = 0:                                                              05418
         BEGIN                                                          05419
         hinc _ hinc0*256;                                              05420
         vinc _ vinc0*256;                                              05421
         END;                                                           05422
      = 1:                                                              05423
         BEGIN                                                          05424
         hinc _ hinc1*256;                                              05425
         vinc _ vinc1*256;                                              05426
         END;                                                           05427
      = 2:                                                              05428
         BEGIN                                                          05429
         hinc _ hinc2*256;                                              05430
         vinc _ vinc2*256;                                              05431
         END;                                                           05432
      = 3:                                                              05433
         BEGIN                                                          05434
         hinc _ hinc3*256;                                              05435
         vinc _ vinc3*256;                                              05436
         END;                                                           05437
      ENDCASE err($"illegal character size");                           05438
   da.dacsize _ da.danocs _ da.dasgcs _ charsize;                       05439
   da.daind _                                                           05440
      MIN((da.daind/da.dahinc)*hinc, da.damind);                        05441
   da.dahinc _ da.danohi _ da.dasghi _ hinc;                            05442
   da.davinc _ vinc;                                                    05443
                                                                        05444
   %% *** Update this from NIC-NLS ***%%                               05445
                                                                        05446
   %%deallocate old da and get new one%%                                05447
      IF nldevice NOT= devlproc THEN                                    05448
         BEGIN                                                          05449
         r1 _ da.dahandle;                                              05450
         IF NOT SKIP !JSYS dda THEN                                     05451
            err($"DDA JSYS failed, qgdf");                              05452
         da.dahandle _ 0;                                               05453
         IF r1 _ da.dalhandle := 0 THEN                                 05454
            BEGIN                                                       05455
            r2 _ linkcnsl;                                              05456
            IF NOT SKIP !JSYS ndda THEN                                 05457
               dismes(2, $"Link NDDA JSYS failed, qgdf");               05458
            END;                                                        05459
         END;                                                           05460
      alocda(&da);                                                      05461
   RETURN;                                                              05462
   END.                                                                 05463
%                                                                       05762
```

```
%show%                                                                  02202
   (xshow) %Execute show Command%                                       02203
      PROCEDURE                                                         02204
         %FORMALS%                                                      02205
            (result, %result record%                                    02206
            parsemode,       %parsing, backup, cleanup%                 02207
            entity,  %entity type%                                      02208
            param,   %param pointer%                                    02209
            dopt);   %directory options for show directory%             02210
         LOCAL                                                          02212
            rhostn, dskcnt,                                             03633
            % stuff for show directory %                                02213
               info, % record saying what was requested %               02214
               gropk,        % record saying how to group things %
                                                                        02215
               sortk % record saying how to wort things %               02216
            ;                                                           02217
         LOCAL STRING filstr[200];                                      03634
         REF                                                            02227
            result, entity, param, dopt;                                02228
      %----------------------%                                         02229
      CASE parsemode OF                                                 02230
         = parsing:                                                     02231
            BEGIN                                                       02232
            *lit* _ NULL;  %used for building status messages%          02233
            CASE entity OF                                              02234
               = 63 %- archive -%: %directory%                          02235
                  cshoarcdir(&param, &param+d2sel, $lit);               02236
               = 0 %- directory -%:                                     02237
                  BEGIN                                                 02238
                  info _ gropk _ sortk _ 0;                             02239
                  *filstr* _ *dirsfname*;                               03635
                  xdiropt( &param, dopt, $info,                         02240
                     $gropk, $sortk, $rhostn, $filstr);                 02241
                  cshodir(info, gropk, sortk, rhostn, $filstr);         02243
                  END;                                                  02245
               = 125 %- disk -%: %space status% %uses connected
               directory%                                               02246
                  IF (dskcnt _ cshodskspa($lit)) > 0 THEN               02247
                     BEGIN                                              03650
                     !gjinf();                                          03652
                     gdname( r2, $filstr);                              03653
                     *filstr* _                                         03654
                        *filstr*, " OVER ALLOCATION BY ",
                        STRING(dskcnt), " PAGES!";                      03656
                     dismes(2, $filstr);                                03657
                     END;                                               03651
               = 15 %- file -%: %status%                                02248
                  CASE param OF                                         02249
                     = 71 %- status -%: %all%                           02250
                        cshofilsta(7, lcfile(), $lit);                  02251
                     = 126 %- default -%: %dir for links%               02252
                        cshofilsta(4, lcfile(), $lit);                  02253
                     = 22 %- marker -%: %list%                          02254
                        cshomarfil(lcfile(), $lit);                     02255
                     = 96 %- modifications -%: %status%                 02256
```

```
                        cshofilsta(?, lcfile(), $lit);                  02257
                    = 37 %- return -%: %ring%                           02258
                        cshofrring(lda(), $lit);                        02259
                    = 89 %- size -%:                                    02260
                        cshofilsta(1, lcfile(), $lit);                  02261
                    ENDCASE err(notyet);                                02262
                = 37 %- return -%: %ring status%                        04341
                    cshosrring( lda(), $lit );                          04342
                = 18 %- name -%: %delimiters for statement%             02263
                    cshonsta(param, $lit);                              02264
                = 112 %- viewspecs -%:                                  02265
                    xshoviespe(lda(), $lit);                            02266
                ENDCASE err(notyet);                                    02267
            CASE entity OF                                              02268
                = 9 %- directory -%: NULL;  %does its own display%      02269

                ENDCASE fbctl( typecalit, $lit );                       02270
            dpset(dspno, endfil, endfil, endfil);                       02271
            END;                                                        02272
        ENDCASE;                                                        02273
    RETURN(&result);                                                    02274
    END.                                                                02275

(xshoviespe)  PROCEDURE (da, string);                                   02276
    LOCAL vs[2];                                                        03150
    REF da, string;                                                     02277
    vs _ da.davspec;                                                    02278
    vs[1] _ da.davspc2;                                                 03151
    curvsp($vs, &string);                                               03152
    RETURN;                                                             02279
    END.                                                                02280

% utility for show directory - prints one file %                        02281
    (xdlpnt)        % print a file for show directory %                  02282
        PROCEDURE                                                       02283
            (function,          % requested function %                 02284
            astr                % address of string to be printed %
                                                                        02285
                                                                        02286
            );                                                          02287
        REF astr;                                                       02288

        CASE function OF                                                02289
            = typenulllit:                                              02290
                fbctl( typenulllit, &astr );                            02291
            = fbaddlit:                                                 02292
                fbctl( fbaddlit, &astr );                               02293
            = typecalit:                                                02294
                fbctl( addcalit, &astr );                               02295
            ENDCASE;                                                    02296
        RETURN;                                                         02297

        END.                                                            02298

                                                                        02299
                                                                        02320
%sort%                                                                  02321
    (xsort) %Execute Sort Command%                                      02322
        PROCEDURE
```

```
      %FORMALS%                                                          02323
         (result, %result record%                                        02324
         parsemode,       %parsing, backup, cleanup%                    02325
         entity,  %entity type%                                          02326
         destination); %destination pointer%                             02327
         REF                                                             02328
            result, entity, destination;                                 02329
      %----------------------%                                           02330
      CASE parsemode OF                                                  02331
         = parsing:                                                      02332
            CASE entity OF                                               02333
               = 2 %- group -%, = 3 %- plex -%:                          02334
                  BEGIN                                                  02335
                  csorgro(&destination, &destination+d2sel);             02336
                  curmkr _ gethed(destination);  curmkr[1] _ 1;          02337
                  dpset(dspstrc, destination, endfil, endfil);           02338
                  END;                                                   02339
               = 1 %- branch -%:                                         02340
                  BEGIN                                                  02341
                  IF (destination := getsub(destination)) =
                  destination THEN err($"Illegal Sort");                 02342
                  destination[d2sel] _ getail(destination);              02343
                  REPEAT CASE(2 %+ group +%);                            02344
                  END;                                                   02345
               ENDCASE err(notyet);                                      02346
         ENDCASE;                                                        02347
      RETURN(&result);                                                   02348
      END.
                                                                         02349
%split%                                                                  02350
%stop%                                                                   02387
   (xstop) %Execute Stop Command%                                        05484
      PROCEDURE                                                          05485
         %FORMALS%                                                       05486
            (result, %result record%                                     05487
            parsemode);      %parsing, backup, cleanup%                  05488
            REF                                                          05489
               result;                                                   05490
      %----------------------%                                           05491
      CASE parsemode OF                                                  05492
         = parsing:                                                      05493
            BEGIN                                                        05494
            recrunfil _ FALSE;                                           05495
            ctlquit();  %ctlquit is also called from HALT%               05496
            END;                                                         05497
         ENDCASE;                                                        05498
      RETURN(&result);                                                   05499
      END.
                                                                         05500
%substitute%                                                             02402
   (xsubstitute) %Execute Substitute Command%                            02403
      PROCEDURE                                                          02404
         %FORMALS%                                                       02405
            (result, %result record%                                     02406
            parsemode,       %parsing, backup, cleanup%                  02407
            textentity,      %text entity type%                          02408
```

```
        structentity, %structure entity type%                          02409
        destination,     %destination pointer%                         02410
        pairs,   %address of pairs list%                               02411
        filterflag,      %TRUE if filter requested%                    02412
        vs);             %viewspecs for filter%                        02413
        REF                                                            02414
           result, structentity, destination, textentity, pairs,
           filterflag, vs;                                             02415
     LOCAL save1, save2, savca, savus, da, adstr[40];  REF da;
                                                                       02416
    %----------------------%                                          02417
    CASE parsemode OF                                                  02418
       = parsing:                                                      02419
        BEGIN                                                          02420
        &da _ lda();                                                   02421
        curmkr _ destination;  curmkr[1] _ 1;                          02422
        noclrall _ TRUE;  %clear window only, not screen; checked
        by clearda%                                                    05635
        save1 _ da.davspec;                                            02423
        save2 _ da.davspc2;                                            02424
        savca _ da.dacacode;                                           03857
        savus _ da.dausqcod;                                           03858
        ON SIGNAL ELSE                                                 03859
           BEGIN                                                       03860
           da.davspec _ save1;                                         03862
           da.davspc2 _ save2;                                         03863
           da.dacacode _ savca;                                        03864
           da.dausqcod _ savus;                                        03865
           END;                                                        03861
        IF filterflag THEN                                             02425
           BEGIN                                                       02426
           da.davspec _ vs.vs1;                                        02427
           da.davspc2 _ vs.vs2;                                        02428
           da.dacacode _ vs.vscacode;                                  03868
           da.dausqcod _ vs.vsusqcod;                                  03869
           END                                                         02429
        ELSE                                                           03870
           BEGIN                                                       03871
           da.davspec _ 0;                                             03878
           da.davspec.vslev _ da.davspec.vstrnc _                      03872
              da.davspec.vsnamf _ da.davspec.vsdaft _                  03879
                 da.davspec.vsindf _ -1;                               03880
           da.dacacode _ da.dausqcod _ 0;                              03874
           END;                                                        03876
        CASE structentity OF                                           02430
           = 4 %- statement -%:                                        02431
              BEGIN                                                    02432
              dpset(dsprfmt, destination, endfil, destination);
                                                                       02433
              clist(ctcmk, destination.stfile, nofile);                02434
              csubsta(destination, pairs, &da);                        02435
              clupdt();                                                02436
              END;                                                     02437
           = 2 %- group -%, = 3 %- plex -%, = 1 %- branch -%:
                                                                       02438
              BEGIN                                                    02439
```

```
                dpset(dspyes,destination,endfil,endfil);        02440
                clist(ctcmk, destination.stfile, nofile);       02441
                csubgro(destination, [&destination+d2sel], pairs,
                &da);                                           02442
                clupdt();                                       02443
                END;                                            02444
             ENDCASE err(notyet);                               02445
          da.davspec _ save1;                                   02446
          da.davspc2 _ save2;                                   02447
          da.dacacode _ savca;                                  03866
          da.dauscod _ savus;                                   03867
          END;                                                  02448
       ENDCASE;                                                 02449
    RETURN(&result);                                            02450
    END.

                                                                02451
%transpose%                                                     02452
   (xtranspose) %Execute Transpose Command%                     02453
      PROCEDURE                                                 02454
         %FORMALS%                                              02455
            (result, %result record%                            02456
            parsemode,        %parsing, backup, cleanup%        02457
            sourcentity,      %source entity type%              02458
            source,  %source pointer%                           02459
            destentity,       %destination entity type%         02460
            destination,      %destination pointer%             02461
            filterflag,       %if TRUE, filtered with viewspecs in vs%
                                                                02462
            vs);              %viewspec string%                 02463
            REF                                                 02464
               result, sourcentity, source, destentity, destination,
               filterflag, vs;                                  02465
         LOCAL adstr[40];  % block for parsing links %          03322
      %----------------------%                                  02466
      CASE parsemode OF                                         02467
         = parsing:                                             02468
            BEGIN                                               03300
            CASE sourcentity OF                                 03302
               = 8 %- link -%:                                  03303
                  BEGIN                                         03305
                  IF source.stastr THEN                         03454
                     BEGIN                                      03455
                     IF NOT FIND                                03456
                        SF(source) $(SP/TAB) ("(/"</"--") THEN  03457
                           ST source _ "<, SF(source) SE(source);
                                                                03458
                     IF NOT FIND                                03459
                        SE(source) $(SP/TAB) (")/">) THEN       03460
                           ST source _ SF(source) SE(source), ">;
                                                                03461
                     END;                                       03462
                  lnkprs( &source, $adstr);                     03306
                  source _ adstr[ls];                           03307
                     source[1] _ adstr[ls+1];                   03318
                  [&source+d2sel] _ adstr[le];                  03308
                     [&source+d2sel+1] _ adstr[le+1];           03319
```

```
                END;                                                    03309
              ENDCASE;                                                  03304
            CASE destentity OF                                          03310
              = 8 %- link -%:                                           03311
                BEGIN                                                   03312
                IF destination.stastr THEN                              03463
                  BEGIN                                                 03464
                  IF NOT FIND                                           03465
                    SF(destination) $(SP/TAB) ("(/"</"--") THEN
                                                                        03466
                                                                        03467
                      ST destination _
                        "<, SF(destination) SE(destination);
                                                                        03472
                  IF NOT FIND                                           03468
                    SF(destination) $(SP/TAB) (")/">) THEN              03469
                      ST destination _                                  03470
                        SF(destination) SE(destination), ">;
                                                                        03473
                                                                        03471
                  END;                                                  03313
                lnkprs( &destination, $adstr);                          03314
                destination _ adstr[ls];                                03320
                  destination[1] _ adstr[ls+1];                         03315
                r&destination+d2sel] _ adstr[le];                       03321
                  r&destination+d2sel+1] _ adstr[le+1];                 03316
                END;                                                    03317
              ENDCASE;                                                  02469
            CASE sourcentity OF
              = 5 %- character -%, = 7 %- invisible -%, = 12 %- text
              -%, = 14 %- word -%, = 13 %- visible -%, = 11 %-
              number -%, = 8 %- link -%:                                02470
                BEGIN                                                   02471
                clist (ctcmk, destination.stfile, source.stfile);
                                                                        02472
                dpset(dsprfmt, destination, source, endfil);            02473
                curmkr _ source;  curmkr[1] _ source[1];                02474
                ctratex(&destination, &destination+d2sel, &source,
                                                                        02475
                &source+d2sel);                                         02476
                clupdt ();                                              02477
                END;                                                    02478
              = 4 %- statement -%:                                      02479
                BEGIN
                clist (ctcsp, destination.stfile, source.stfile);
                                                                        02480
                dpset(dspstrc, destination, source, endfil);            02481
                curmkr _ destination;  curmkr[1] _ 1;                   02482
                ctrasta(destination, source, filterflag, &vs);
                                                                        02483
                                                                        02484
                clupdt ();                                              02485
                END;
              = 2 %- group -%, = 3 %- plex -%, = 1 %- branch -%:
                                                                        02486
                                                                        02487
                BEGIN
                clist (ctcsp, destination.stfile, source.stfile);
                                                                        02488
                dpset(dspstrc,destination,source,endfil);               02489
                curmkr _ source;  curmkr[1] _ 1;                        02490
```

```
                ctragro(destination, destination[d2sel], source,
                source[d2sel], filterflag, &vs);                    02491
                clupdt ();                                          02492
                END;                                                02493
             ENDCASE err(notyet);                                   02494
          END;                                                      03301
       ENDCASE;                                                     02495
    RETURN(&result);                                                02496
    END.

                                                                    02497
%trim%                                                              02498
   (xtrim) %Execute Trim Command%                                   02499
      PROCEDURE                                                     02500
         %FORMALS%                                                  02501
            (result, %result record%                                02502
            parsemode,        %parsing, backup, cleanup%            02503
            parameter);       %number of versions%                  02504
            REF                                                     02505
               result, parameter;                                   02506
      LOCAL tlngth; % temporary for string length %                 02507
      %----------------------%                                      02508
      CASE parsemode OF                                             02509
         = parsing:                                                 02510
            BEGIN                                                   02511
            result _ 0;                                             02512
            result _ getstring(3000, $dspblk);                      02513
            *[result]* _ "Trimmed Files Are:", CR, LF;              02514
            tlngth _ [result].L;                                    02515
            ctridir(getpint(&parameter, &parameter+d2sel), result);
                                                                    02516
               %trim connected directory%                           02517
            IF ( [result].L > tlngth ) THEN                         02518
               fbctl(typecalit, result)                             02519
            ELSE fbctl (typecalit, $"No Files Trimmed" );           05773
            END;                                                    02521
         = backup, = cleanup:                                       02522
            IF result THEN freestring(result, $dspblk);             02523
         ENDCASE;                                                   02524
      RETURN(&result);                                              02525
      END.

                                                                    02526
%undelete%                                                          02527
   (xundelete) %Execute Undelete Command%                           02528
      PROCEDURE                                                     02529
         %FORMALS%                                                  02530
            (result, %result record%                                02531
            parsemode,        %parsing, backup, cleanup%            02532
            entity,  %entity type%                                  02533
            filename);        %filename pointer%                    02534
            REF                                                     02535
               result, entity, filename;                            02536
         LOCAL stid, tlngth, rhostn;                                02537
         LOCAL STRING filestr[200];                                 03294
      %----------------------%                                      02540
      CASE parsemode OF                                             02541
         = parsing:                                                 02542
```

```
            BEGIN                                                          02543
            result _ 0;                                                    02544
            CASE entity OF                                                 02545
               = 15 %- file -%:                                            02546
                  BEGIN                                                    02547
                  result _ getstring(3000, $dspblk);                       02548
                  *[result]* _ "Undeleted Files Are:", CR, LF;             02549
                  tlngth _ [result].L;                                     02550
                  rhostn _ lnbfls( &filename, 0, $filestr);                03295
                  cundfil(rhostn, $filestr, result);                       02554
                  IF ( [result].L > tlngth ) THEN                          02555
                     fbctl( typecalit, result)                             02556
                  ELSE fbctl( typecalit, $"No Files Undeleted" );          02557
                                                                           02558
                  END;                                                     02559
               = 63 %- archive -%: %file%                                  02560
                  cundarcfil(&filename, &filename+d2sel);                  02561
               = 06 %- modifications -%: %to file%                         02562
                  BEGIN                                                    02563
                  stid _ orgstid;                                          02564
                  stid.stfile _ lcfile();                                  02565
                  clist (ctlcfm, stid.stfile, nofile);                     02566
                  dpset(dspallf, stid, endfil, endfil);                    02567
                  cundmodfil(stid.stfile);                                 02568
                  <IOEXEC, unlkclist> (); %check clist items%              02569
                  clupdt ();                                               02570
                  END;                                                     02571
               ENDCASE err(notyet);                                        02572
            END;                                                           02573
         = backup, = cleanup:                                              02574
            IF result THEN freestring(result, $dspblk);                    02575
         ENDCASE;                                                          02576
      RETURN(&result);                                                     02577
      END.                                                                 02578

%update%                                                                   02579
   (xupdate) %Execute Update Command%                                      02580
      PROCEDURE                                                            02581
         %FORMALS%                                                         02582
            (result, %result record%                                       02583
            parsemode,        %parsing, backup, cleanup%                   02584
            entity,  %entity type%                                         02585
            filename);        %filename pointer%                           02586
         LOCAL tp2, fileno, stid;                                          02589
         LOCAL STRING filstring[200];                                      02590
         REF                                                               02591
            result, entity, filename, tp2;                                 02592
         %----------------------%                                         02593
         CASE parsemode OF                                                 02594
            = parsing:                                                     02595
               BEGIN                                                       03807
               stid _ orgstid;                                             03808
               stid.stfile _  fileno _ lcfile();                           03809
               dpset(dsprfmt, stid, endfil, endfil);                       02598
               CASE entity OF                                              02599
                  = 127 %- old -%:
```

```
                cupdfil (fileno, oldversion, 0);                    02600
             = 128 %- new -%:                                       02601
                cupdfil (fileno, newversion, 0);                    02602
             = 129 %- compact -%: %file%                            02603
                BEGIN                                               04497
                dpset(dspyes, stid, endfil, endfil);                04496
                clist(15, fileno, 0);                               02597
                cupdfil (fileno, upcompact, 0);                     02604
                END;                                                04498
             = 130 %- rename -%: %file%                             02605
                BEGIN                                               02606
                % move file name to local string %                  02607
                   CASE lnbfls( &filename, 0, $filstring) OF        03289
                      = lhostn: NULL;                               03291
                      ENDCASE                                       03292
                         err($"Remote File Manipulations Not
                         Implemented Yet");                         03293
                cupdfil (fileno, newname, $filstring);              02616
                END;                                                02617
             ENDCASE err(notyet);                                   02618
          *filstring* _ NULL;                                       02623
          filnam( fileno, $filstring);                              02624
          dismes( 2, $filstring);                                   02625
          END;                                                      02626
       ENDCASE;                                                     02627
    RETURN(&result);                                                02628
    END.
                                                                    02629
%verify%                                                            02630
   (xverify) %Execute Verify Command%                               02631
      PROCEDURE                                                     02632
         %FORMALS%                                                  02633
            (result, %result record%                                02634
            parsemode);      %parsing, backup, cleanup%             02635
            REF                                                     02636
               result;                                              02637
      %----------------------%                                      02638
      CASE parsemode OF                                             02639
         = parsing:                                                 02640
            BEGIN                                                   04482
            cverfil(lcfile(), TRUE);                                02641
            dismes(1, $"Successful: internal structure is OK");     04484
            END;                                                    04483
         ENDCASE;                                                   02642
      RETURN(&result);                                              02643
      END.
                                                                    02644
%TAB%                                                               02645
   (xtab) PROCEDURE( % Execute repeat last search command %         02646
      % FORMAL ARGUMENTS %                                         02647
         resultptr,  % ptr to result record %                       02648
         parsemode); % interpreter parsing mode %                   02649
         REF resultptr;                                             02650
      LOCAL da;  REF da;                                            02651
      LOCAL TEXT POINTER t1, t2;                                    02652
      LOCAL STRING treps[2003];                                     02653
```

```
                                                                        02654
      CASE parsemode OF                                                 02655
         = parsing:                                                     02656
            BEGIN                                                       02657
            CASE srchtype OF                                            02658
               = wordtyp:                                               03167
                  *treps* _ "=w";                                       03170
               = wordls:                                                03171
                  *treps* _ "=ws";                                      02663
               = contnt:                                                03168
                  *treps* _ "=c";                                       02668
               = contls:                                                03169
                  *treps* _ "=cs";
               ENDCASE err($"<tab> valid only to repeat a previous
               search");                                                02670
            *treps* _ ".n", '"', *conreg*, '"', *treps*;                02671
            dismes(1, $treps);                                          02672
            FIND SF(*treps*) ^t1 SE(*treps*) ^t2;                       02674
            &da _ cspupdate _ lda();                                    05757
            curmkr _ da.dacsp;  curmkr[1] _ da.daccnt;                  05758
            caddexp($t1, $t2, &da, $curmkr);                            02675
            dpset(dspjpf, curmkr, endfil, endfil);                      02677
            dismes(0,0);                                                03185
            END;                                                        02678
         ENDCASE;                                                       02679
      RETURN (&resultptr);                                              02680
      END.                                                              02681

                                                                        02706
%<LF>%                                                                  02707
   (xlinefeed) PROCEDURE( % Execute TNLS print next statement %         02708
      % FORMAL ARGUMENTS %                                              02709
         resultptr,  % ptr to result record %                           02710
         parsemode); % interpreter parsing mode %                       02711
         REF resultptr;                                                 02712
      LOCAL stid, da;  REF da;                                          02713
      LOCAL lvs[2];  % viewspec param record %                          02714
      LOCAL TEXT POINTER t1, t2;                                        02715
      LOCAL STRING string[5];                                           02716
      %----------%                                                      02717
      CASE parsemode OF                                                 02718
         = parsing:                                                     02719
            BEGIN                                                       05745
            &da _ lda();
            IF da.daempty THEN err($"No file currently loaded.");       05746

            *string* _".n";                                             02720
            FIND SF(*string*) ^t1 SE(*string*) ^t2;                     02721
            caddexp($t1, $t2, &da, $curmkr);                            02722
            lvs _ da.davspec;                                           02723
            lvs[1] _ da.davspc2;                                        02724
            cprista( curmkr, $lvs, &da );                               02725
            dpset(dspjpf, curmkr, endfil, endfil);                      02726
            END;                                                        02727
         ENDCASE;                                                       02728
      RETURN (&resultptr);                                              02729
      END.                                                              02730
```

```
%^%                                                                    02731
   (xuparrow) PROCEDURE( % Execute TNLS print previous statement %
                                                                       02732
      % FORMAL ARGUMENTS %                                           02733
        resultptr,  % ptr to result record %                         02734
        parsemode); % interpreter parsing mode %                     02735
        REF resultptr;                                               02736
      LOCAL stid, da;  REF da;                                       02737
      LOCAL lvs[27];  % viewspec param record %                      02738
      LOCAL TEXT POINTER t1, t2;                                     02739
      LOCAL STRING string[50];                                       02740
      %----------%                                                   02741
      CASE parsemode OF                                              02742
         = parsing:                                                  02743
           BEGIN                                                     02744
           &da _ lda();                                              05748
           IF da.daempty THEN err($"No file currently loaded.");
                                                                     05749
           *string* _".b";                                           02745
           FIND SF(*string*) ^t1 SE(*string*) ^t2;                   02746
           caddexp($t1, $t2, &da, $curmkr);                          02747
           lvs _ da.davspec;                                         02748
           lvs[17] _ da.davspc2;                                     02749
           cprista( curmkr, $lvs, &da );                             02750
           dpset(dspjpf, curmkr, endfil, endfil);                    02751
           END;                                                      02752
      ENDCASE;                                                       02753
      RETURN (&resultptr);                                           02754
      END.
                                                                     02755
%.%                                                                  02756
   (xperiod) PROCEDURE( % Execute TNLS print current location %      02757
      % FORMAL ARGUMENTS %                                           02758
        resultptr,  % ptr to result record %                         02759
        parsemode); % interpreter parsing mode %                     02760
        REF resultptr;                                               02761
      LOCAL STRING astrng[50]; % collection string %                 02762
      LOCAL da;  REF da;                                             05752
      %----------%                                                   02763
      CASE parsemode OF                                              02764
         = parsing:                                                  02765
           BEGIN                                                     02766
           &da _ lda();                                              05750
           IF da.daempty THEN err($"No file currently loaded.");
                                                                     05751
           cspupdate _ FALSE;                                        02767
           ccurloc($curmkr, $astrng );                               02768
           typeas( $astrng );                                        02769
           END;                                                      02770
      ENDCASE;                                                       02771
      RETURN (&resultptr);                                           02772
      END.
                                                                     02773
%/%                                                                  02774
   (xslash) PROCEDURE( % Execute TNLS / command %                    02775
      % FORMAL ARGUMENTS %                                           02776
```

```
        resultptr,  % ptr to result record %                          02777
        parsemode); % interpreter parsing mode %                      02778
        REF resultptr;                                                02779
      LOCAL STRING string[100];                                       02780
      LOCAL da;  REF da;                                              05753
      CASE parsemode OF                                               02781
        = parsing:                                                    02782
          BEGIN                                                       02783
          &da _ lda();                                                05754
          IF da.daempty THEN err($"No file currently loaded.");
                                                                      05755
          cspupdate _ FALSE;                                          02784
          ccurcon( $curmkr, $string );                                02785
          typeas($string);                                            02786
          END;                                                        02787
        ENDCASE;                                                      02788
      RETURN (&resultptr);                                            02789
      END.
                                                                      02790
%\%                                                                   02791
  (xbslash) PROCEDURE( % Execute TNLS \ command %                     02792
      % FORMAL ARGUMENTS %                                            02793
        resultptr,  % ptr to result record %                          02794
        parsemode); % interpreter parsing mode %                      02795
        REF resultptr;                                                02796
      LOCAL da, csp, cnt;                                             02797
      LOCAL lvs[2];  % viewspec param record %                        02798
      REF da;                                                         02799
      CASE parsemode OF                                               02800
        = parsing:                                                    02801
          BEGIN                                                       02802
          &da _ lda();                                                02805
          IF da.daempty THEN err($"No file currently loaded.");
                                                                      05756
          cspupdate _ FALSE;                                          02803
          csp _ curmkr;  cnt _ curmkr[1];                             02804
          lvs _ da.davspec;                                           02806
          lvs[1] _ da.davspc2;                                        02807
          cprista( curmkr, $lvs, &da );                               02808
          curmkr _ csp;  curmkr[1] _ cnt;                             02809
          END;                                                        02810
        ENDCASE;                                                      02811
      RETURN (&resultptr);                                            02812
      END.
                                                                      02813
% Substitute support routines %                                       02941
  (subsinit) PROC( % initializes state for the substitute command %
                                                                      02942
      % FORMAL ARGUMENTS %                                           02943
        resultptr,      % ptr to result record %                     02944
        parsemode,      % parsing mode %                             02945
        textent);       % text entity type for the substitute %      02946
      LOCAL % VARIABLES %                                             02947
        subdsp,         % ptr to substitute display buffer %         02948
        ttype; % text entity code %                                   02949
      LOCAL STRING subm[30];  % string for "Subs = " message %        02950
```

```
    REF subdsp, resultptr, textent;                                    02951
    % ---------- %                                                     02952
    CASE parsemode OF                                                  02953
       = parsing:                                                      02954
          BEGIN                                                        02955
          % set ttype according to type of text entity-- This code
          should be removed after sbinit is changed to expect the new
          type of text entity codes %                                  02956
             ttype _ CASE textent OF                                   02957
                = 5 %- character -%:  charv;                           02958
                = 14 %- word -%:              wordv;                   02959
                = 11 %- number -%:            numbrv;                  02960
                = 13 %- visible -%:           visv;                    02961
                = 7 %- invisible -%:  invisv;                          02962
                = 8 %- link -%:               linkv;                   02963
                = 12 %- text -%:              textv;                   02964
                ENDCASE err( notyet );                                 02965
          % allocate a display buffer for the substitute processor %
                                                                       02966
             resultptr _ resultptr[1] _ 0; %will be used in cleanup. %
                                                                       03702
             resultptr _ &subdsp _ getstring( 640, $dspblk );          02967
          % allocate an astr buffer for the substitute processor %
                                                                       03699
             resultptr[1] _ getstring( 500, $dspblk );                 03698
          % initialize the substitute state info %                    02968
             sbinit( $subhed, $subrec, &subdsp, resultptr[1], ttype );
                                                                       02969
          % initialize substitutions count %                          02977
             subcnt _ 0;                                               02978
          END;                                                         02979
       = backup,                                                       02980
       = cleanup:                                                      02981
          BEGIN                                                        02982
          % deallocate the display buffer if allocated %              02983
             IF resultptr THEN freestring( resultptr, $dspblk );       02984
          % deallocate the astr if allocated %                        03700
             IF resultptr[1] THEN freestring( resultptr[1], $dspblk );
                                                                       03701
          % display the number of substitutions made %                02985
             *subm* _ "Substitutions made: ", STRING(subcnt);          02986
             dismes(1, $subm);                                         02987
          END;                                                         02993
       ENDCASE;                                                        02994
    RETURN (&resultptr ); % TRUE return %                              02995
    END.
                                                                       02996
(sbinit) PROCEDURE(sbhed, sbrec, sbdsp, sbastr, sbttype);              02997
    % initialize substitute data structure %                          02998
    LOCAL j;                                                           02999
    POINTER sbhed;                                                     03000
    sbhed.sbrp _ sbrec;                                                03001
    sbhed.sbdp _ sbdsp;                                                03002
    sbhed.sbas _ sbastr;                                               03003
    sbhed.sbtt _ sbttype;                                              03004
    *[sbastr]* _ NULL;                                                 03005
```

```
    FOR i _ 0 UP UNTIL >=2008 DO [sbdsp+j] _ 0;                          03006
    RETURN;                                                              03007
    END.
                                                                         03008
                                                                         03009
(sub1dsp) PROCEDURE( % substitute prompting function %                   03010
    % FORMAL ARGUMENTS %                                                 03011
       resultptr,      % ptr to the result record %                     03012
       parsemode,      % parsing mode %                                 03013
       textent,        % text entity type for the substitute %          03014
       newflag);       % TRUE if new text entity is being collected %
                                                                         03015
    LOCAL STRING string[50];  % prompting string %                       03016
    REF resultptr, textent, newflag;                                     03017
    % ------------ %                                                    03018
    CASE parsemode OF                                                    03019
       = parsing:                                                        03020
          BEGIN                                                          03021
          IF nlmode = typewriter THEN crlf();                            03022
          IF newflag                                                     03023
             THEN *string* _ "New "                                      03024
             ELSE *string* _ "for all occurrences of old ";              03025
          *string* _ *string*, *[[&textent + 1]]*;                       03026
          fbctl( echostr, $string );                                     03027
          END;                                                           03028
       ENDCASE;                                                          03029
    RETURN (&resultptr);                                                 03030
    END.
                                                                         03031
(sub2dsp) PROCEDURE( % substitute prompting function %                   03032
    % FORMAL ARGUMENTS %                                                 03033
       resultptr,      % ptr to the result record %                     03034
       parsemode,      % parsing mode %                                 03035
       type,           % type of selection made %                       03324
       tptr); % ptr to string tptr record %                              03036
    LOCAL tptr2, adstr[40];                                              03037
    LOCAL STRING string[100]; % prompting string %                       03038
    REF resultptr, tptr, tptr2, type;                                    03039
    % ------------ %                                                    03040
    CASE parsemode OF                                                    03041
       = parsing:                                                        03042
          BEGIN                                                          03043
          CASE type OF                                                   03325
             = 8 %- link -%:                                             03326
                BEGIN                                                    03327
                IF (tptr.stastr) AND                                     03337
                   ( NOT FIND SF(tptr) $(SP/TAB) ("(/"</"--") ) THEN
                                                                         03338
                      ST tptr _ "(, SF(tptr) SE(tptr);                   03339
                IF (tptr.stastr) AND                                     03340
                   ( NOT FIND SE(tptr) $(SP/TAB) (")/">) ) THEN          03341
                      ST tptr _ SF(tptr) SF(tptr), ">;                   03342
                lnkprs( &tptr, $adstr);                                  03328
                tptr _ adstr[1s];                                        03329
                   tptr[1] _ adstr[1s+1];                                03330
                [&tptr+d2sel] _ adstr[le];                               03331
```

```
                [&tptr+d2sel+1] _ adstr[ie+1];                          03332
              END;                                                      03333
           ENDCASE;                                                     03334
        IF n!mode = fulldisplay                                         03044
           AND NOT tptr.stastr THEN                                     03045
              BEGIN                                                     03046
              &tptr2 _ &tptr + d2sel;                                   03047
              *string* _ tptr tptr2;                                    03048
              aplit( $string ); % feedback bugged param %               03049
              END;                                                      03050
        END;                                                            03051
     ENDCASE;                                                           03052
  RETURN (&resultptr);                                                  03053
  END.                                                                  03054

(substatus) PROCEDURE (                                                 05658

  % The status of a substitute command is printed out before
  execution, given the entity type being substituted%                   05659
  % FORMAL ARGUMENTS %                                                  05660
     resultptr,      % ptr to the result record %                        05661
     parsemode,      % parsing mode %                                    05662
     type);          % type of selection made %                          05663
  LOCAL pairstr, cardp, wbp1, wbp2, i, j, colwidth;                     05664
  LOCAL STRING between[10], statstr[500], sentstr[10], tempstr[120],
  oldstr[120], newstr[120];                                             05665
  POINTER cardp;                                                        05666
  REF pairstr, type, resultptr;                                         05667

                                                                        05668
  CASE parsemode OF                                                     05669
     = parsing:                                                         05670
        BEGIN                                                           05671
        colwidth _ 20;  %width of a column for non-text
        substitutions%                                                  05672
        %Set up the table heading for output%                           05673
           *sentstr* _ *[[&type+1]]*;                                   05674
           FOR j _ sentstr.L UP UNTIL = sentstr.M DO                    05675
              *sentstr* _ *sentstr*,SP;                                 05676
           *statstr* _ "   SUBSTITUTE ", *sentstr*, CR, LF;             05677
           IF *sentstr* = "TEXT      " THEN                            05678
              *between* _ CR, LF, "OLD:  "                              05679
           ELSE                                                         05680
              BEGIN                                                     05681
              *statstr* _ *statstr*, "NEW ", *sentstr*, "               OLD
              ", *sentstr*, CR, LF, CR, LF;                             05682
              *between* _ "  ";                                         05683
              END;                                                      05684

        %Get ahold of the packed string of pairs%                       05685
        &pairstr _ subhed.sbas;                                         05686
                                                                        05687
        i _ 1;                %Index through the string%                05688
        WHILE i < pairstr.L DO                                          05689
           BEGIN                                                        05690
           oldstr.L _ newstr.L _ cardp _ 0;                             05691
```

```
        cardp _ subhed.sbdp + *pairstr*[i];                                 05692
           %get pointer to chain describing entities staring with
           the first character of this entity%                              05693
        IF [cardp] = 0 THEN err($"substatus:  Substitution array
        bad.")                                                              05694
        ELSE cardp _ [cardp];                                               05695
        DO                                                                  05696
           BEGIN                                                            05697
           wbp1 _ chbmty + $oldstr;                                         05698
           wbp2 _ cardp.catbp;                                              05699
           FOR j _ 0 UP UNTIL = cardp.catnc DO                              05700
              BEGIN                                                         05701
              IF j = oldstr.M THEN EXIT;                                    05702
              ^wbp1 _ ^wbp2;     %store the old string%                     05703
              END;                                                          05704
           oldstr.L _ MIN(cardp.catnc, j);                                  05705
           *tempstr* _ *pairstr*[i TO oldstr.L+i-1];                        05706
           (comstrs):                                                       05707
           IF *tempstr* = *oldstr* THEN                                     05708
              BEGIN      %This card describes the next pair%
                                                                            05709
              wbp1 _ chbmty + $newstr;                                      05710
              wbp2 _ cardp.carbp;                                           05711
              FOR j _ 0 UP UNTIL = cardp.carnc DO                           05712
                 BEGIN                                                      05713
                 IF j = newstr.M THEN EXIT;                                 05714
                 ^wbp1 _ ^wbp2;  %store the new string%                     05715
                 END;                                                       05716
              newstr.L _ MIN(cardp.carnc, j);                               05717
              EXIT;                                                         05718
              END                                                           05719
           ELSE cardp _ cardp.canxt;     %next entry%                       05720
           END                                                              05721
        WHILE cardp;                                                        05722
                                                                            05723
        IF oldstr.L = 0 THEN err($"substatus:  String not found
        in array.");                                                        05724
                                                                            05725
        %Add this entry to output status string%                            05726
        FOR j _ newstr.L UP UNTIL >= colwidth DO                            05727
           *newstr* _ *newstr*, SP;                                         05728
        IF *sentstr* = "TEXT      " THEN *statstr* _ *statstr*,
        CR, LF, "NEW:  ";  %blank lines between pairs of text%
                                                                            05729
        *statstr* _ *statstr*, *newstr*, *between*, *oldstr*, CR,
        LF;                                                                 05730
                                                                            05731
        %Increment i for next pair of strings%                              05732
        i _ i + cardp.catnc + cardp.carnc;                                  05733
        END;                                                                05734

     %Send it out and set display globals%                                  05735
     fbctl(typecalit, $statstr);                                            05736
     dpset(dsrno, endfil, endfil, endfil);                                  05737

     END;                                                                   05738
```

```
      ENDCASE;                                                          05739

      RETURN( &resultptr );                                             05740
                                                                        05741
   END.                                                                 05742
(subpsave) PROCEDURE( % stashes away parameters for substitute %
                                                                        03055
   % FORMAL ARGUMENTS %                                                03056
      resultptr,      % ptr to the result record %                     03057
      parsemode,      % parsing mode %                                 03058
      entity,         % ptr to entity type for parsing link %          05597
      newptr,         % ptr to the new entity %                        03059
      oldptr);        % ptr to the old entity %                        03060
   LOCAL new2, old2, adstr[40];                                         03061
   LOCAL STRING newstr[200], oldstr[200];                               03062
   REF resultptr, newptr, oldptr, new2, old2, entity;                   03063
   % ------------ %                                                    03064
   CASE parsemode OF                                                    03065
      = parsing:                                                        03066
         BEGIN                                                          03067
         CASE entity OF                                                 05598
            = 8 %- link -%:                                             05599
               BEGIN                                                    05600
               lnkprs( &newptr, $adstr);                                05601
               newptr _ adstr[ls];                                      05602
                  newptr[1] _ adstr[ls+1];                              05603
               [&newptr+d2sel] _ adstr[le];                             05604
                  [&newptr+d2sel+1] _ adstr[le+1];                      05605
               lnkprs( &oldptr, $adstr);                                05606
               oldptr _ adstr[ls];                                      05607
                  oldptr[1] _ adstr[ls+1];                              05608
               [&oldptr+d2sel] _ adstr[le];                             05609
                  [&oldptr+d2sel+1] _ adstr[le+1];                      05610
               END;                                                     05611
            ENDCASE;                                                    05612
         % fetch parameters to local strings %                          03068
            &new2 _ &newptr + d2sel;                                    03069
            &old2 _ &oldptr + d2sel;                                    03070
            *newstr* _ newptr new2;                                     03071
            *oldstr* _ oldptr old2;                                     03072
         % check length of the old (target) string %                    03073
            IF oldstr.L = empty THEN                                    03074
               err( $"cannot substitute for a null text field" );
                                                                        03075
         % stash away the parameter strings %                           03076
            sbpush( $newstr, $oldstr, $subhed );                        03077
         % set return value to be pointer to saved pairs %             03078
            resultptr _ $subhed;                                        03079
         END;                                                           03080
      ENDCASE;                                                          03081
   RETURN (&resultptr);                                                 03082
   END.
                                                                        03083
(sbpush) PROCEDURE(str1, str2, hed);                                    03084
   %.....Documentation.....%                                            03085
```

```
      %The test strings are sorted by initial character and chained
      together, with the head of the chain in a character-code
      indexed array subdsp.  This allows a very fast check whether a
      character can possibly begin a test string.  Each
      test-replacement pair is represented by a record (card) which
      is linked to others for the same initial test character through
      the canxt field.%                                              03086
      %Global variables used:                                        03087
         subcnt  count of substitutions                              03088
         lit     A-string for building up new statement              03089
         swork   internal work area, see (stbpget)                   03090
      %                                                              03091
   % add pair to substitute list %                                   03092
   LOCAL astr, len, len1, len2, subrp, asa, cap;                     03093
   POINTER hed, subrp, cap;                                          03094
   REF str1, str2, astr;                                             03095
   &astr _ hed.sbas;                                                 03096
   len _ astr.L;                                                     03097
   asa _ &astr + 1; %origin of text for A-string%                    03098
   len1 _ str1.L;                                                    03099
   len2 _ str2.L;                                                    03100
   *astr* _ *astr*, *str2*;                                          03101
   IF hed.sbtt = numbrv THEN                                         03102
      % pad replacement if shorter %                                 03103
      FOR len1 UP UNTIL >=len2 DO                                    03104
         *astr* _ *astr*, SP;                                        03105
   *astr* _ *astr*, *str1*;                                          03106
   subrp _ hed.sbrp;                                                 03107
   subrp.carnc _ len1;                                               03108
   subrp.catnc _ len2;                                               03109
   subrp.carbp _ conbp(asa,len+len2);                                03110
   subrp.catbp _ conbp(asa,len);                                     03111
   subrp.canxt _ 0;                                                  03112
   %link card into appropriate chain%                                03113
      cap _ hed.sbdp + *str2*[1];                                    03114
      IF [cap] = 0 THEN %empty chain%                                03115
         [cap] _ subrp                                               03116
      ELSE BEGIN %link on end%                                       03117
         cap _ [cap];                                                03118
         WHILE cap.canxt DO cap _ cap.canxt;                         03119
         cap.canxt _ subrp;                                          03120
      END;                                                           03121
   hed.sbrp _ subrp + lcard;                                         03122
   RETURN END.                                                       03123
                                                                     03124
DECLARE bpary = (0106777777777B, 350788, 260788, 170788, 100788);
                                                                     03125
(conbp) PROCEDURE(base,cn);                                          03126
   % construct byte pointer assuming 5 characters per word.  cn is
   the character number %                                            03127
   LOCAL q, r;                                                       03128
   DIV cn / 5, q, r;                                                 03129
   RETURN(base+bpary[r]+q);                                          03130
   END.                                                              03131
                                                                     03132
FINISH of psedit                                                     03133
```

PSHELP

```
PSHELP >>> < NLS, PSHELP.NLS;30, >, 10-FEB-77 17:10 KIRK ;;;;
FILE pshelp  % L10 <REL-NLS>PSHELP.REL %% (L10,) (rel-nls,pshelp.rel,) %
                                                                    02526
   % L10 <REL-NLS>PSHELPTEST.REL % % (L10,) (rel-nls,pshelptest.rel,) %
                                                                    02986
% DECLARATIONS %                                                    02527
   % ********** %                                                  02985
      DECLARE EXTERNAL helpfirstfile, menusw;                       02982
      REF menusw;                                                   03014
   REF inpt;                                                        02528
   REF conrng, qda, qsw, curstk, qstorblk, qnewstmt, hlpcmdstk;     02529
% HELP Parsefunctions %                                            02530
   (lookback) PROCEDURE (resultptr, parsemode, string);             02531
      % parsing function which looks at the next input char.  If it is
      an "<" or "_" character, then it returns TRUE, otherwise it
      returns FALSE %                                               02532
         REF resultptr, string;                                     02534
         CASE parsemode OF                                          02535
            = parsing:                                              02536
               CASE lookc() OF                                      02537
                  = '_, = '<:                                       02538
                     inpt();                                        02539
                  ENDCASE RETURN (FALSE);                           02540
            = parsehelp:                                            02541
               *string* _ "<";                                      02542
            = parseqmark:                                           02543
               BEGIN                                                02544
               *string* _ NULL;                                     02545
               RETURN;                                              02546
               END;                                                 02547
            ENDCASE;                                                02548
         RETURN (&resultptr);                                       02549
         END.
                                                                    02550
   (lkup) PROCEDURE (resultptr, parsemode, string);                 02560
      % parsing function which looks at the next input char.  If it is a
      ^ character, then it returns TRUE, otherwise it returns FALSE %
                                                                    02561
         REF resultptr, string;                                     02563
         CASE parsemode OF                                          02564
            = parsing:                                              02565
               CASE lookc() OF                                      02566
                  = '^:                                             02567
                     inpt();                                        02568
                  ENDCASE RETURN (FALSE);                           02569
            ENDCASE;                                                02570
         RETURN (&resultptr);                                       02571
         END.
                                                                    02572
   (dumprompt) PROCEDURE (curptr, parsemode, string);               02573
      % ? %                                                         02574
         REF curptr, string;                                        02575
         CASE parsemode OF                                          02576
            = parsing:                                              02577
               NULL;                                                02578
            = parsehelp:                                            02579
```

```
            *string* _ "T";                                          02581
         = parseqmark:                                               02582
            BEGIN                                                    02583
            *string* _ "T";                                          02585
            RETURN;                                                  02586
            END;                                                     02587
      ENDCASE;                                                       02588
      RETURN (&curptr);                                              02589
      END.
                                                                     02590
(looknxt) PROCEDURE (resultptr, parsemode, string);                  02591
   % parsing function which looks at the next input char.  If it is a
   REPEAT character, then it reads the char and returns TRUE,
   otherwise it returns FALSE %                                      02592
      REF resultptr, string;                                         02594
      CASE parsemode OF                                              02595
         = parsing:                                                  02596
            CASE lookc() OF                                          02597
               = rptchar:                                            02598
                  NULL;                                              02599
               ENDCASE RETURN (FALSE);                               02600
         = parsehelp:                                                02601
            *string* _ "RPT:";                                       02602
         = parseqmark:                                               02603
            BEGIN                                                    02604
            *string* _ "REPEAT";                                     02605
            RETURN;                                                  02606
            END;                                                     02607
         ENDCASE;                                                    02608
      RETURN (&resultptr );                                          02609
      END.
                                                                     02610
(mylookbug) PROCEDURE (curptr, parsemode, string);                   02611
   % lookbug looks at the next inpt character, if it is a CA, then a
   true return is taken else FALSE is returned %                     02612
      REF curptr, string;                                            02613
      CASE parsemode OF                                              02614
         = parsing:                                                  02615
            CASE lookc() OF                                          02616
               = cachar, = rptchar, = inschar:                       02617
                  NULL;                                              02618
            ENDCASE RETURN (FALSE);                                  02619
         = parsehelp:                                                02620
            IF nlmode = fulldisplay THEN *string* _ "B:"             02621
            ELSE *string* _ "OK:";                                   02622
         = parseqmark:                                               02623
            BEGIN                                                    02624
            IF nlmode = fulldisplay THEN *string* _ "BUG"            02625
            ELSE *string* _ "OK";                                    02626
            RETURN;                                                  02627
            END;                                                     02628
      ENDCASE;                                                       02629
      RETURN (&curptr);                                              02630
      END.
                                                                     02631
(myrdconfirm) PROCEDURE (curptr, parsemode, string);                 02632
```

```
   % readconfirm looks at the next inpt character, if it is a
   CA/REPEAT/INSERT, then it is read and a true return is taken else
   FALSE is returned %                                                  02633
      REF curptr, string;                                               02634
      CASE parsemode OF                                                 02635
         = parsing:                                                     02636
            CASE lookc() OF                                             02637
               = cachar, = rptchar, = inschar:                          02638
                  BEGIN                                                 02639
                  % stick ptr to castring into result record %          02640
                     curptr _ $castr;                                   02641
                  % read over the CA %                                  02642
                     inpt();                                            02643
                  END;                                                  02644
            ENDCASE RETURN (FALSE);                                     02645
         = parsehelp:                                                   02646
            *string* _ "OK";                                            02647
         = parseqmark:                                                  02648
            BEGIN                                                       02649
            *string* _ "OK or node";                                    02650
            RETURN;                                                     02651
            END;                                                        02652
      ENDCASE;                                                          02653
      RETURN (&curptr);                                                 02654
      END.                                                              02655

(sethflg) PROCEDURE (resultptr, parsemode);                             02551
   % Sets athelp flag for questionmark > <nls, parser, fbhelp> %        02552

      REF resultptr, string;                                            02553
      CASE parsemode OF                                                 02554
         = parsing: athelp _ TRUE;                                      02555
         = cleanup, =backup: athelp _ FALSE;                            02556
         ENDCASE;                                                       02557
      RETURN (&resultptr);                                              02558
      END.                                                              02559

(checkmore) PROCEDURE (result, parsemode);                              02876
   % dav: checks the value of the global variable 'moremenu' %          02877
      CASE parsemode OF                                                 02878
         = parsing :                                                    02879
            BEGIN                                                       02880
            IF moremenu THEN                                            02881
               BEGIN                                                    02882
               RETURN(result);                                          02883
               END;                                                     02884
            RETURN(0);                                                  02885
            END;                                                        02886
         ENDCASE;                                                       02887
      RETURN(result);                                                   02888
      END.                                                              02889

(rstmor) PROCEDURE (result, parsemode);                                 02890
   % Reinitialize moremenu to FALSE; close sequences. %                 02891
      CASE parsemode OF                                                 02892
         = parsing :                                                    02893
```

```
            BEGIN                                                   02894
            moremenu _ FALSE;                                       02895
            IF &qsw THEN moreterm();                                02896
               % Must be at a lower level because of symbol
               conflicts. Closes sequences and resets viewspecs. %
                                                                    03015
            END;                                                    02900
         ENDCASE;                                                   02901
      RETURN(result);                                               02902
      END.
                                                                    02903
% Help execution functions %                                       02656
   (hlpinit) PROCEDURE (resultptr, parsemode, entrymode, intparm); 02657
      % This procedure does the initialization required upon entry into
      the help command.  If the entrymode is CNTLQ, then a control-Q was
      typed to get in: this implies that the command data structure has
      been set up and may be followed to find the first item to be
      displayed.  If the entry mode is HLPCOM, then we open files and
      allocate appropriate storage.  In cleanup or backup modes we
      release storage and sequences and set up for the next entry. %
                                                                    02658
      LOCAL                                                         02659
         i,                                                         02660
         entryptr, % pointer to subsystem stack; used to get pointer
         to name of subsystem %                                     02661
         helpad, % work space %                                     02662
         typead, % work space %                                     02663
         shortcut,                                                  02664
         % **********                                               02988
            entstd,                                                 02665
            topstd,                                                 02666
            ********** %                                            02989
         end,                                                       02667
         da;                                                        02668
      LOCAL TEXT POINTER ptr1, ptr2, filptr;                        02669
      LOCAL STRING toolnm[100], dirstr[40], namstr[1000];           02670
      REF resultptr, entrymode, entryptr, da, intparm, hlpcmdstk,
      qda, qsw, qstorblk, qnewstmt;                                 02671
      % ********** %                                               03016
         REF menusw;                                                03017
      CASE parsemode OF                                             02672
         = parsing:                                                 02673
            BEGIN                                                   02674
            inhelp _ qagain := 0;                                   02675
               % qagain set ONLY here and in helphlp.  Necessary
               because inhelp is reset upon termination which happens
               when control q is hit again! %                       02978
            IF hdebug THEN *dirstr* _ "XHELP,"                      02676
            ELSE *dirstr* _ "HELP,";                                02979
            IF inhelp THEN                                          02677
               % We are in the help command already.  Show the help
               branch. %                                            02679
               BEGIN                                                02678
               % **********                                         03008
                  *namstr* _ *dirstr*, "CORE,HELP";                 03010
                  ********** %                                      03009
```

```
            *namstr* _ *dirstr*, "NLS,HELP";                    02680
         helpad _ $namstr;                                      02681
         typead _ 18 % + name + % ;                             02682
         RETURN (helpshow(&resultptr, parsing, $typead,
         $helpad));                                             02683
         END;                                                   02684
      shortcut _ FALSE;                                         02685
      IF entrymode = 179 % + hlpcom + % THEN                    02686
         BEGIN % set up address for help command for this tool.
         %                                                      02687
         % Get storage for name stack %                        02688
            IF NOT &hlpcmdstk THEN                              02689
               &hlpcmdstk _ getarray(hpcmdmax + 1, $dspblk);
                                                                02981
         % get name of this tool %                             02690
            &entryptr _ $sbstack + sbstkx - $sbentsize;         02691
            *toolnm* _ *[entryptr.sbnptr]*;                     02692
         IF NOT intparm THEN namstr.L _ 0                       02693
            % No initial parameter has been specified %        02694
         ELSE  % user has specified a search %                  02695
            BEGIN                                               02696
            ptr1 _ intparm; ptr1[1] _ intparm[1];               02697
            ptr2 _ intparm[d2sel]; ptr2[1] _ intparm[d2sel+1];
                                                                02698
            *namstr* _ ptr1 ptr2;                               02699
            END;                                                02700
         END                                                    02701
      ELSE % user typed ctrl-q %                                02702
         BEGIN                                                  02703
         IF hlpcmdstk THEN                                      02704
            BEGIN                                               02705
            % change destination array for cntlq mode to an
            address.  Assume the first word is the name of the
            tool. %                                             02706
            *toolnm* _ *[hlpcmdstk[1]]*;                        02707
            IF hlpcmdstk=1 THEN                                 02708
               *namstr* _ "0" % just go to origin %             02709
            ELSE                                                02710
               BEGIN                                            02711
               namstr.L_0;                                      02712
               FOR i_2 UP UNTIL > hlpcmdstk DO                  02713
                  *namstr* _ *namstr*, SP, *[hlpcmdstk[i]]* ;
                                                                02714
               END                                              02715
            END                                                 02716
         ELSE                                                   02717
            *namstr* _ "no-parse-info-from-CLI";                02718
         END;                                                   02719
      % open current tool's file to initialize for first search
      %                                                         02720
         IF NOT (hlpfileno _ clohfil($toolnm)) THEN             02721
            IF NOT (hlpfileno _ clohfil($"NLS")) THEN           02722
               IF intparm AND FIND SF(*namstr*) [',] THEN       02723
                  % initialize hlpfileno to current file %
                                                                02724
                  hlpfileno _ cda.dacsp.stfile                  02725
```

```
            ELSE  % the user did not specify a file %                02726
               err(1,$"No help for this tool");                      02727
      % ********** %                                                 02984
         helpfirstfile _ hlpfileno;                                  02983
            % remember the file in which we started %                02987
      ON SIGNAL ELSE                                                 02728
         BEGIN                                                       02729
         ON SIGNAL ELSE;                                             02730
         err($"I am the elusive Help command bug:
         tell FEEDBACK details of what you were doing.");            02731
         END;                                                        02732
      % General initialization %                                     02733
         &qda _ &qsw _ &qstorblk _ &qnewstmt _ 0;                    02734
         % ********** %                                             03011
            &menusw _ 0;                                             03012
         % Redirect control-Q for use in help. %                     02735
            chntab[24] _ $helphlp .V 186;                            02736
         % Set flag saying we're in Help command. %                  02737
            inhelp _ TRUE;                                           02738
         % Initialize global error flag %                           02739
            hseqer _ FALSE;                                          02740
         % Save old da's (and jump stacks) and get a new one
         for use by query; upon quitting, these will be
         restored %                                                  02741
            IF nlmode = fulldisplay THEN                             02742
               BEGIN                                                 02743
               clearda(0);                                           02744
               clrall( 0, TRUE );                                    02745
               % Save away auxiliary bit for calculator display
               area. %                                               02746
                  calcaux _ cda.daauxiliary ;                        02747
               % set bit in da's so that they are ignored. %
                                                                     02748
                  end _ (&da _ $dpyarea) + dacnt*dal;                02749
                  DO da.daauxiliary _ TRUE UNTIL                     02750
                     (&da _ &da + dal) >= end;                       02751
               END;                                                  02752
            &qda _ newda();                                          02753
            intdafl(&qda);                                           02754
            IF nlmode = fulldisplay THEN                             02755
               BEGIN                                                 02756
               alocda(&qda);                                         02757
               dpset(dspno, endfil, endfil, endfil);                 02758
               END;                                                  02759
         % The sequence generator for query/help. %                  02760
            qda.dafrzl _ 0;                                          02761
            qda.davspec _ defvs1;                                    02762
            qda.davspc2 _ defvs2;                                    02763
            qda.dausqcod _ $queryseq;                                02764
            qda.davspec.vsusqf _ TRUE;                               02765
            qda.davspec.vsrind _ TRUE;                               02766
            qda.davspec.vsindf _ FALSE;                              02767
            qda.davspec.vspagf _ FALSE;                              02768
            qda.davspec.vsbrof _ TRUE;                               02769
            qdavspc _ qda.davspec;                                   02770
            qdavs2 _ qda.davspc2;                                    02771
```

```
        % Initialize pre-search pointers %                          02772
          % **********                                              02990
             topstd _ entstd _ qda.dacsp _ orgstid;                 02993
             ********** %                                          02991
             qda.dacsp _ orgstid;                                   02773
          qda.daccnt _ 1;                                           02774
          qda.dacsp.stfile _ hlpfileno;                             02775
        % Initialize the query stack areas %                       02776
          % **********                                              02994
             qstrinit( entstd, topstd );                            02995
             ********** %                                          02996
             qstrinit( orgstid, orgstid );                          02997
          curindex _ entind;                                        02778
          confre _ 0;                                               02779
          newstk _ TRUE;                                            02780
        % release space for hlpcmdstk %                            02781
          freestring(&hlpcmdstk := 0, $dspblk);                     02782
      % Print out the first node. %                                02783
        typead _ 18 % + name + % ;                                  02784
        helpad _ $namstr;                                           02785
        RETURN( helpshow( &resultptr, parsing, $typead,
        $helpad));                                                  02786
      END;                                                          02787
    ENDCASE                                                         02788
      % Termination of HELP command; release storage if
      necessary. %                                                  03007
      BEGIN                                                         02789
      IF qagain THEN RETURN(&resultptr);                            02790
      % redirect control-Q interrupt %                             02791
        chntab[24] _ $gotohelp .V 1B6;                              02792
      % Reset help flag %                                          02793
        inhelp _ FALSE;                                             02794
      IF &qsw THEN closeseq(&qsw := 0);                             02795
      % ********** %                                               03018
        IF &menusw THEN closeseq(&menusw := 0);                     03019
      IF &qnewstmt  THEN freestring(&qnewstmt :=0, $dspblk);
                                                                    02796
      IF &qstorblk  THEN freestring(&qstorblk :=0, $dspblk);
                                                                    02797
      IF &hlpcmdstk THEN freestring(&hlpcmdstk:=0, $dspblk);
                                                                    02798
                                                                    02799
      IF &qda THEN                                                  02800
        BEGIN                                                       02801
        IF nlmode = fulldisplay THEN                                02802
          BEGIN                                                     02803
          clearda(&qda);                                            02804
          delda(&qda);
          % set bit in da's so that they are not ignored. %        02805
            end _ (&da _ $dpyarea) + dacnt*dal;                     02806
            DO da.daauxiliary _ FALSE UNTIL                         02807
              (&da _ &da + dal) >= end;                             02808
          % Restore auxiliary bit for calculator display
          area. %                                                   02809
            cda.daauxiliary _ calcaux ;                             02810
          dpset(dspallf, endfil, endfil, endfil);                   02811
```

```
                    recred();                                          02812
                    END                                                02813
                  ELSE delda(&qda);                                    02814
                  END;                                                 02815
                dismes(0);                                             02816
                END;                                                   02817
        RETURN(&resultptr);                                            02818
        END.                                                           02819

(helpshow) PROCEDURE (resultptr, parsemode, type, param);              02820
    % ? %                                                              02821
        LOCAL stid;                                                    02822
        LOCAL STRING emptmes[300];                                     02823
        REF resultptr, type, param;                                    02824
        % ********** %                                                03000
           REF qda, qsw;                                               03001
        CASE parsemode OF                                              02825
           = parsing:                                                  02826
              BEGIN                                                    02827
              CASE type OF                                             02828
                 = 28:  % up   - find the source of the current node %
                                                                       02829
                                                                       02830
                   BEGIN
                   stid _ gda.dacsp _ qgetup(gda.dacsp, &qda,
                                                                       02831
                   $emptmes);                                          02832
                   <conint>(stid);  % update context stack %           02833
                   <chkdsp>(stid);  % display node %                   02834
                   RETURN(&resultptr);                                 02835
                   END;
                = 18:  % name - param contains a pointer to a string
                with  either a number (menu) or a word to be used in a
                                                                       02836
                name search %                                          02837
                   BEGIN
                   IF NOT qsearch(param.stpsid % string address % ,
                                                                       02838
                      curindex : stid, curindex) THEN                  03004
                      BEGIN                                            02839
                      *emptmes* _ *[param.stpsid]*, "?";               02840
                      dismes(1,$emptmes);                              02841
                      RETURN(&resultptr);                              02842
                      END;                                             02843
                   <conint>(stid);                                     02844
                   <chkdsp>(stid);                                     02845
                   RETURN(&resultptr);                                 02846
                   END;                                                02847
                = 34:  % next - show the rest of the menu that would
                                                                       02848
                not fit %                                              02849
                   BEGIN                                               02850
                   IF NOT &qsw THEN
                      dismes(2, $"Last menu was complete: No more")
                                                                       02853
                   ELSE <chkdsp>(0);   % Call qdisp in continue mode %
                                                                       02855
                                                                       02856
                   RETURN (&resultptr);                                02857
                   END;
                = 33:  % back - param contains the stid of the back
```

```
                node to be displayed. %                                 02858
                   BEGIN                                                02859
                   curindex _ param;                                    02860
                   stid     _ param[1];                                 02861
                   % **********                                         03002
                      newstk _ FALSE;                                   02862
                      qda.dacsp _ stid;                                 02863
                      qda.daccnt _ 1;                                   02864
                      dismes(0);                                        02865
                      <chkdsp>(stid);                                   02866
                      ********** %                                      03003
                      dismes(0);                                        03005
                      <conint>(stid);                                   02999
                      <chkdsp>(stid);                                   03006
                   RETURN(&resultptr);                                  02867
                   END;                                                 02868
                = 171: % menu - param contains the stid of a menu item
                %                                                       02869
                   typeas($"notyet");                                   02870
                ENDCASE err($"Help system error:
                Invalid param passed to helpshow");                     02871
             END;                                                       02872
          ENDCASE;                                                      02873
       RETURN(&resultptr);                                              02874
       END.                                                             02875

(xhlpring) PROCEDURE (resultptr, parsemode, nwindex);                   02904
   % provides feedback for stepping through the rings for BACK
   command in help %                                                    02905
      LOCAL conad, stid, stdb, len;                                     02906
      LOCAL TEXT POINTER tp1, tp2;                                      02907
      LOCAL STRING temp[20];                                            02908
      REF resultptr, nwindex, conad;                                    02909
      CASE parsemode OF                                                 02910
         = parsing:                                                     02911
            BEGIN                                                       02912
            % advance through the ring %                                02913
            % If we are at the beginning, say we can go no further. %
                                                                        02914
                                                                        02915
              IF nwindex = 0 THEN                                       02916
                 BEGIN                                                  02917
                 !sti(18M,CA);                                          02918
                 resultptr _ curindex;                                  02919
                 resultptr[1] _                                         02920
                    IF curindex = entind THEN entcon                    02921
                    ELSE [&conrng + curindex*rnglnt];                   02922
                 dismes( 2, $"No others have been shown");              02923
                 RETURN(&resultptr);                                    02924
                 END;                                                   02925
            &conad _                                                    02926
               IF nwindex = entind THEN $entcon                         02927
               ELSE &conrng + nwindex*rnglnt;                           02928
            IF nwindex # entind THEN                                    02929
               BEGIN                                                    02930
               nwindex _ conbck(nwindex);                               02931
               &conad _
```

```
                    IF nwindex = entind THEN $entcon                    02932
                    ELSE &conrng + nwindex*rngInt;                      02933
                 END;                                                   02934
              stid _ conad;                                             02935
              % display the current statement %                         02936
                 % display first 20 chars of stmt in name area %        02937
                    % get statement length %                            02938
                       IF NOT lodprop( stid, txttyp : stdb) THEN        02939
                          err($"No text block associated with node");
                                                                        02940
                       len _ [stdb].schars + 1; % number of chars %
                                                                        02941
                    % construct text ptrs to each end of stmt %        02942
                       tp1 _ stid;                                      02943
                       tp1[1] _ 1;                                      02944
                       IF NOT FIND SF(tp1) [EOL ^tp2 _tp2 /  "##" ^tp2
                       _2tp2 ] THEN                                     02945
                          BEGIN                                         02946
                          tp2 _ stid;                                   02947
                          tp2[1] _ len;                                 02948
                          END;                                          02949
                       % Truncate %                                     02950
                          tp2[1] _ MIN(20, tp2[1]);                     02951
                    % assign something to the string "temp" %         02952
                       IF tp2[1] > 1 THEN *temp* _ tp1 tp2 ELSE *temp*
                       _ "<NULL>";                                      02953
                 dn($temp); % display string %                          02954
              % save the current state in the result record %          02955
                 resultptr _ nwindex;                                   02956
                 resultptr[1] _ stid;                                   02957
              END;                                                      02958
           = backup,                                                    02959
           = cleanup:                                                   02960
              dn( $""); % clear the name area %                         02961
           ENDCASE;                                                     02962
        RETURN(&resultptr);                                             02963
        END.
                                                                        02964
(qbkint) PROCEDURE (resultptr, parsemode);                              02965
                                                                        02966
   % ? %                                                                02967
        REF resultptr;                                                  02968
        CASE parsemode OF                                               02969
           = parsing:                                                   02970
              BEGIN                                                     02971
              resultptr _ curindex;                                     02972
              END;                                                      02973
           ENDCASE;                                                     02974
        RETURN(&resultptr);
        END.
                                                                        02975
FINISH of pshelp                                                        02976
```

PSIDENT

```
< NLS, PSIDENT.NLS;4, >, 9-SEP-74 16:29 CHI ;;;;( NLS, PSIDENT.NLS;4, ),
15-MAY-74 16:30 KEV ;
FILE psident % L10 <rel-nls>psident %% (L10,) (rel-nls,psident.rel,) %
                                                                      02
% Parser-support for IDENTIFICATION subsystem %                       03
(xiloaidfil)    % load Master Ident-File file %
PROCEDURE (resultptr, parsemode);                                     04
   REF resultptr;                                                     06
   LOCAL STRING infostr[10000];                                       07
   CASE parsemode OF                                                  08
      = parsing:                                                      09
         BEGIN                                                       010
         IF NOT idfno THEN                                           012
            BEGIN                                                    013
            idfno _ loaidfil();                                     0914
            idmodflag _ idmodified _ FALSE;                         0821
            IF ckiwheel THEN                                         015
               BEGIN                                                 016
               ckiwheel _ FALSE;                                     017
               identwheel _ FALSE;                                   018
               IF NOT ckident($initsr, $infostr, idfno) THEN         019
                  err($"Illegal user IDENT");                        020
               IF geticapabilities($infostr, $infostr, 0,0) THEN     021
                  IF FIND SF(*infostr*) ["Ident-Wheel"] THEN         022
                     identwheel _ TRUE;                              023
               END;                                                  024
            END;                                                    0916
         IF idcrec THEN freestring(idcrec, $dspblk);                 026
         idcrec _ getstring(2000, $dspblk);                          027
         IF idcident THEN freestring(idcident, $dspblk);             028
         idcident _ getstring(20, $dspblk);                          029
         idcrtype _ 0;                                               030
         END;                                                        031
   ENDCASE;                                                          032
   RETURN( &resultptr );                                             033
   END.
                                                                     034
(xicloidfil)    % close Master Ident-File file %
PROCEDURE (resultptr, parsemode);                                    035
   REF resultptr;                                                    037
   CASE parsemode OF                                                 038
      = parsing:                                                     039
         BEGIN                                                       040
         IF idfno THEN                                               042
            BEGIN                                                    043
            [flntadr(idfno)].flnoclos _ FALSE;                      0471
            close(idfno);                                            044
            idfno _ 0;                                               045
            IF idcrec THEN freestring(idcrec := 0, $dspblk);         046
            IF idcident THEN freestring(idcident := 0, $dspblk);     047
            END;                                                     048
         END;                                                        049
      ENDCASE;                                                       050
   RETURN( &resultptr );                                             051
   END.
                                                                     052
```

```
(xiupdate)      % update modified record %
PROCEDURE (resultptr, parsemode, now);                                  053
   LOCAL STRING badids[100], workstr[100];                             0488
   REF resultptr, now;                                                  055
   CASE parsemode OF                                                    056
      = parsing:                                                        057
         BEGIN                                                          058
         IF NOT idcrec OR NOT [idcrec].L THEN err($"no record loaded!");
                                                                       0895
         IF now THEN                                                   0472
            dismes(1, $"Updating Master Ident-File.  Please wait.")     059
         ELSE                                                          0473
            dismes(1, $"Partially Updating Master Ident-File.  Please
            wait.");                                                   0474
         %check for presence of necessary fields%                       061
            %ident, name, coord, org, mail address%                    0487
               IF NOT [idcident].L THEN                                0490
                  err($"Ident is missing--update not performed");      0493
               %ident if not new record%                               0579
                  IF NOT nwrecflag THEN                                0580
                     BEGIN                                             0581
                     getiid ( idcrec, $workstr, 0,0);                  0582
                        %don't use idcident since if it is different
                        then getiid, it is new%                        0583
                     IF NOT oldid($workstr, idfno) THEN                0584
                        BEGIN                                          0585
                        *badids* _ "Bad Ident in (old) IDENT field: ",
                        *badids*, " -- Update not performed";          0586
                        err ( $badids );                               0587
                        END;                                           0588
                     END;                                              0589
               getinam ( idcrec, $workstr, 0,0);                       0497
               IF NOT workstr.L THEN                                   0498
                  err($"name must be presant--update not performed");
                                                                       0499
               getiadd ( idcrec, $workstr, 0,0);                       0500
               IF NOT workstr.L THEN                                   0501
                  err($"Address must be presant--update not performed");
                                                                       0502
               IF idcrtype =  indtyp THEN                              0506
                  BEGIN  %organization%                                0507
                  getiorg ( idcrec, $workstr, 0,0);                    0503
                  IF NOT workstr.L THEN                                0504
                     err($"Organization must be presant--update not
                     performed");                                      0505
                  IF NOT oldid($workstr, idfno) THEN                   0529
                     BEGIN                                             0530
                     *badids* _ "Bad Ident in ORGANIZATION field: ",
                     *badids*, " -- Update not performed";             0531
                     err ( $badids );                                  0532
                     END;                                              0533
                  END                                                  0508
               ELSE                                                    0509
                  BEGIN  %coordinator%                                 0510
                  geticord ( idcrec, $workstr, 0,0);                   0511
                  IF NOT workstr.L THEN                                0512
```

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

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

      END.                                                              075

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

```
            ENDCASE %use shoifield%                                          0592
               BEGIN                                                         0593
               *string* _ *[idcrec]*;                                        0598
               *[idcrec]* _ *recstring*;                                     0599
               *recstring* _ *string*;                                       0600
               *string* _ *[idcident]*;                                      0601
               *[idcident]* _ *idstring*;                                    0602
               *idstring* _ *string*;                                        0603
               ON SIGNAL ELSE                                                0597
                  BEGIN                                                      0610
                  *[idcrec]* _ *recstring*;                                  0607
                  *[idcident]* _ *idstring*;                                 0608
                  END;                                                       0609
               shoifield (&resultptr, parsemode, &fieldname);                0604
               *[idcrec]* _ *recstring*;                                     0605
               *[idcident]* _ *idstring*;                                    0606
               END;                                                          0594
         END;                                                                0458
      END;                                                                    092
   = backup, = cleanup: shoifield(&resultptr, parsemode, 0);                  0869
   ENDCASE;                                                                   093
RETURN( &resultptr );                                                         094
END.
                                                                              095
(newrec)        %initialize current record, for adding new records to
ident-file%
PROCEDURE (resultptr, parsemode);                                             096
   REF resultptr;                                                             098
   CASE parsemode OF                                                          099
      = parsing:                                                              0100
         BEGIN                                                                0101
         nwrecflag _ idmodflag _ idmodified _ TRUE;                           0102
         *[idcrec]* _ "()


         ";                                                                   0103
         *[idcident]* _ NULL;                                                 0104
         END;                                                                 0105
   ENDCASE;                                                                   0106
   RETURN( &resultptr );                                                      0107
   END.
                                                                              0108
(isnewrec)     %is the current record a new record%
PROCEDURE (resultptr, parsemode);                                             0109
   REF resultptr;                                                             0111
   CASE parsemode OF                                                          0112
      = parsing:                                                              0113
         RETURN( IF nwrecflag THEN &resultptr ELSE 0 );                       0114
   ENDCASE;                                                                   0115
   RETURN( &resultptr );                                                      0116
   END.
                                                                              0117
(canmodify)     %is the user allowed to modify this record?%
PROCEDURE (resultptr, parsemode);                                             0118
   REF resultptr;                                                             0120
   CASE parsemode OF                                                          0121
```

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

```
            FIND SF(*namestr*);                                          0655
            *newid* _ READC;                                             0657
            LOOP CASE char _ READC OF                                    0658
               = SP:                                                     0659
                  BEGIN                                                  0664
                  CASE char _ READC OF                                   0675
                     = PT:   *newid* _ *newid*, char;                    0676
                     = ENDCHR: EXIT LOOP;                                0677
                     ENDCASE REPEAT CASE;                                0678
                  END;                                                   0665
               = '-: *newid* _ *newid*, char;                            0662
               = ENDCHR: EXIT LOOP;                                      0674
               ENDCASE;                                                  0660
            END;                                                         0649
         ENDCASE                                                         0650
            err ($"Illegal record type discovered in ASSTENTID");
                                                                         0673
      FOR i_0 UP UNTIL >= 9 DO %now check uniguenesss and modify by
      adding digits%                                                     0618
         IF oldid($newid, idfno) THEN                                    0621
            gnxtid($newid, idfno)                                        0622
         ELSE EXIT LOOP;                                                 0667
      setiid(idcrec, $newid);                                            0133
      *[idcident]* _ *newid*;                                            0134
      END;                                                               0136
ENDCASE;                                                                 0137
RETURN( &resultptr );                                                    0138
END.
                                                                         0139
(noident)       %is there no ident assigned to current record%
PROCEDURE (resultptr, parsemode);                                        0140
   REF resultptr;                                                        0142
   CASE parsemode OF                                                     0143
      = parsing:                                                         0144
         RETURN( IF [idcident].L THEN &resultptr ELSE 0 );               0145
   ENDCASE;                                                              0146
   RETURN( &resultptr );                                                 0147
   END.
                                                                         0148
(isindividual) %is the current record for an individual%
PROCEDURE (resultptr, parsemode);                                        0163
   REF resultptr;                                                        0165
   CASE parsemode OF                                                     0166
      = parsing:                                                         0167
         BEGIN                                                           0900
         IF NOT idcrec OR NOT [idcrec].L THEN err($"no record loaded");
                                                                         0901
         RETURN( IF idcrtype = indtyp THEN &resultptr ELSE FALSE );      0168
         END;                                                            0903
   ENDCASE;                                                              0169
   RETURN( &resultptr );                                                 0170
   END.
                                                                         0171
(okident)       %is the passed ident ok in syntax and uniqueness%
PROCEDURE (resultptr, parsemode, identstr);                              0172
   REF resultptr, identstr;                                              0174
```

```
                                                                    0175
   CASE parsemode OF                                                0176
      = parsing:
         RETURN( IF ( FIND SF(*[idcident]*) -(LD/"-) ) OR
         oldid(idcident, idfno) THEN 0 ELSE &resultptr );           0177
                                                                    0178
   ENDCASE;                                                         0179
   RETURN( &resultptr );
   END.                                                             0180

(checkname)      %check the ident file for individuals with this last
name, print brief status for each%
PROCEDURE (resultptr, parsemode, lastname);                         0181
   LOCAL TEXT POINTER z1;                                           0885
   LOCAL STRING idstr[50], namestr[50];                             0886
   REF resultptr, lastname;                                         0183
   CASE parsemode OF                                                0184
      = parsing:                                                    0185
         BEGIN                                                      0888
         z1 _ lastname[d2sel];  z1[1] _ lastname[d2sel+1];          0889
         *namestr* _ lastname z1;                                   0890
         RETURN( IF namesearch ($namestr, $idstr, lname, idfno) THEN
         &resultptr ELSE 0 );                                       0186
         END;                                                       0891
   ENDCASE;                                                         0187
   RETURN( &resultptr );                                            0188
   END.                                                             0189

(xiaddmem)       % add members to membership list %                 0190
PROCEDURE (resultptr, parsemode, fieldname, idlist);                0192
   REF resultptr, fieldname, idlist;                                0689
   LOCAL TEXT POINTER z1, z2, z3, z4;                               0687
   LOCAL STRING
      workstr[500], idstring[50], newidlist[500], commstr[200];     0706
   CASE parsemode OF                                                0193
      = parsing:                                                    0194
         BEGIN                                                      0195
         IF NOT idcrec OR NOT [idcrec].L THEN err($"no recod loaded");
                                                                    0904
                                                                    0682
         CASE fieldname OF                                          0683
            = $membership: getimem (idcrec, $workstr, 0,0);         0684
            = $groups: getigrps(idcrec, $workstr, 0,0);             0685
            ENDCASE
               err($"Illegal field type encountered by XIADDMEM");  0688
                                                                    0690
         z1 _ idlist[d2sel];  z1[1] _ idlist[d2sel+1];              0691
         *newidlist* _ idlist z1;                                   0692
         FIND SF(*newidlist*) ^z4;                                  0693
         LOOP                                                       0694
            BEGIN
            FIND z4 $(SP/",) ^z1 $(LD/"-) ^z2 $SP ("( ^z3 _z3
            [")/ENDCHR] ^z4 / ^z3 ^z4);                             0695
            *idstring* _ +z1 z2;                                    0696
            *commstr* _ z3 z4;                                      0705
            IF NOT idstring.L THEN EXIT LOOP;                       0697
            FIND SF(*workstr*) ^z1;                                 0700
            WHILE ( FIND z1 [*idstring*] ^z1 ) DO                   0698
               IF FIND z1 < [(SP/",) > CH /ENDCHR] *idstring*
               (SP/",/ENDCHR) THEN                                  0701
```

```
            REPEAT LOOP 2;                                                  0702
         *workstr* _ *idstring*, *commstr*, SP, *workstr*;                  0703
         END;                                                               0704
      CASE fieldname OF                                                     0707
         = $membership: setimem (idcrec, $workstr, 0,0);                    0708
         = $groups: setigrps(idcrec, $workstr, 0,0);                        0709
         ENDCASE                                                            0710
            err($"Illegal field type encountered by XIADDMEM");             0711
      idmodified _ TRUE;                                                    0862
      END;                                                                  0681
ENDCASE;                                                                    0196
RETURN( &resultptr );                                                       0197
END.
                                                                            0198
(xidelmem)      % delete members to membership list %
PROCEDURE (resultptr, parsemode, fieldname, idlist);                        0199
   REF resultptr, fieldname, idlist;                                        0713
   LOCAL TEXT POINTER z1, z2, z3, z4;                                       0714
   LOCAL STRING                                                             0715
      workstr[500], idstring[50], newidlist[500];                           0716
   CASE parsemode OF                                                        0717
      = parsing:                                                            0718
      BEGIN                                                                 0719
      IF NOT idcrec OR NOT [idcrec].L THEN err($"no record loaded");
                                                                            0905
      CASE fieldname OF                                                     0720
         = $membership: getimem (idcrec, $workstr, 0,0);                    0721
         = $groups: getigrps(idcrec, $workstr, 0,0);                        0722
         ENDCASE                                                            0723
            err($"Illegal field type encountered by XIADDMEM");             0724
      z1 _ idlist[d2sel];  z1[1] _ idlist[d2sel+1];                         0725
      *newidlist* _ idlist z1;                                              0726
      FIND SF(*newidlist*) ^z2;                                             0727
      LOOP                                                                  0728
         BEGIN                                                              0729
         FIND z2 $(SP/',) ^z1 $(LD/'-) ^z2;                                 0730
         *idstring* _ +z1 z2;                                               0731
         IF NOT idstring.L THEN EXIT LOOP;                                  0733
         FIND SF(*workstr*) ^z4;                                            0734
         WHILE ( FIND z4 [*idstring*] ^z4 ) DO                              0735
            IF FIND z4 < [(SP/',) > CH /ENDCHR] ^z3 *idstring*
            ($(SP/',)/ENDCHR) ('( ['')/ENDCHR] / ) ^z4 THEN                 0736
               BEGIN                                                        0750
               ST z3 z4 _ NULL;                                             0751
               REPEAT LOOP 2;                                               0737
               END;                                                         0749
         *idstring* _ "Ident not found: ", *idstring*;                      0753
         dismes(2, $idstring);                                              0752
         END;                                                               0739
      CASE fieldname OF                                                     0740
         = $membership: setimem (idcrec, $workstr, 0,0);                    0741
         = $groups: setigrps(idcrec, $workstr, 0,0);                        0742
         ENDCASE                                                            0743
            err($"Illegal field type encountered by XIADDMEM");             0744
      idmodified _ TRUE;                                                    0863
      END;                                                                  0745
```

```
                                                                    0746
   ENDCASE;                                                         0747
   RETURN( &resultptr );
   END.                                                             0748

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

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

```
                                                                    0275
REF resultptr, param;                                               0276
CASE parsemode OF                                                   0277
   = parsing:                                                       0278
      BEGIN                                                         0279
      z1 _ param[d2sel];  z1[1] _ param[d2sel+1];                    0280
      *string* _ +param z1;                                         0281
      IF *string* = "RABBIT" THEN RETURN( &resultptr);              0800
      IF NOT identwheel THEN
         err($"Illegal password supplied.  Your attempt to use this
         priviledged command has been reported to system personell");
                                                                    0801
                                                                    0282
      err($"Illegal password");                                     0283
      END;                                                          0284
 ENDCASE;                                                           0285
 RETURN( &resultptr );
 END.                                                               0286

(xiloadrecord) % load an old record %                               0287
PROCEDURE (resultptr, parsemode, param);                            0289
   LOCAL TEXT POINTER z1;                                           0290
   LOCAL STRING idstring[30];                                       0291
   REF resultptr, param;                                            0292
   CASE parsemode OF                                                0293
      = parsing:                                                    0294
      BEGIN                                                         0295
      z1 _ param;  z1[1] _ param[d2sel+1];                          0296
      *idstring* _ param z1;                                        0297
      *[idcrec]* _ NULL;                                            0298
      IF NOT ckident($idstring, idcrec, idfno) THEN                 0299
         err($"Illegal Ident");                                     0300
      *[idcident]* _ NULL;                                          0301
      getiid(idcrec, idcident, 0,0);                                0302
      nwrecflag _ FALSE;                                            0303
      IF jgrptst(idcrec, 0) THEN idcrtype _ grptyp                  0304
      ELSE                                                          0305
         IF orgtst(idcrec, 0) THEN idcrtype _ orgtyp                0306
         ELSE idcrtype _ indtyp;                                    0813
      CASE idcrtype OF                                              0814
         = indtyp: %individual%
            idmodflag _ IF *[idcident]* = *initsr* THEN TRUE ELSE
                                                                    0815
            FALSE;                                                  0816
         ENDCASE                                                    0817
            BEGIN                                                   0818
            geticord (idcrec, $idstring, 0,0);
            idmodflag _ IF *idstring* = *initsr* THEN TRUE ELSE
                                                                    0819
            FALSE;                                                  0820
            END;                                                    0867
      idmodified _ FALSE;                                           0307
      END;                                                          0308
   ENDCASE;                                                         0309
   RETURN( &resultptr );
   END.                                                             0310

(setifield)     % set specified fieldname to specified value %
PROCEDURE (resultptr, parsemode, fieldname, value, value2, value3); 0311
                                                                    0313
   LOCAL proced;
```

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

```
=$nlshost: %NLS Mail address%  &prcced _ $setinlhost;                    0361
                                                                         0362
=$name:                                                                  0832
   BEGIN                                                                 0831
   &proced _ $setinam;
   IF idcrtype = indtyp THEN %must concatenate names
   togeter%                                                              0834
      BEGIN                                                              0837
      t2 _ value2[d2sel];  t2[1] _ value2[d2sel+1];                      0835
      t3 _ value3[d2sel];  t3[1] _ value3[d2sel+1];                      0836
      *locstr* _ value t1, ", ", value2 t2, SP, value3 t3;
                                                                         0839
      stnamcap ($locstr );                                               0840
      FIND SF(*locstr*) ^value SE(*locstr*) ^t1;                         0841
      END                                                                0838
   END;                                                                  0833
=$membership:  &proced _ $setimem;                                       0363
=$organization:                                                          0364
   BEGIN                                                                 0365
   IF idcrtype NOT= indtyp THEN err($"Only allowed for
   INDIVIDUAL's records");                                               0366
   IF value = $independent THEN                                          0367
      BEGIN                                                              0368
      *locstr* _ "IND";                                                  0369
      FIND SF(*locstr*) ^value SE(*locstr*) ^t1;                         0370
      END;                                                               0371
   &proced _ $setiorg;                                                   0372
   END;                                                                  0373
=$phone:  &proced _ $setiphone;                                          0374
=$rtype: %record type%                                                   0375
   BEGIN                                                                 0376
   idcrtype _ (CASE value OF                                             0377
      = $group: grptyp;                                                  0378
      = $individual: indtyp;                                             0379
      = $organization: orgtyp;                                           0380
      ENDCASE err(notyet) );                                             0381
   RETURN ( &resultptr );                                                0382
   END;                                                                  0383
=$secondary: %organization%                                              0384
   IF idcrtype NOT= indtyp THEN                                          0385
      err($"Only allowed for INDIVIDUAL's records")                      0386
   ELSE &proced _ $setisorg;                                             0387
=$subcollections:  &proced _ $setisubcol;                                0388
=$type: %of organization%                                                0389
   IF idcrtype NOT= orgtyp THEN err($"Only allowed for
   ORGANIZATION's records")                                              0390
   ELSE                                                                  0391
      BEGIN                                                              0843
      &proced _ $setitype;                                               0842
      CASE value OF                                                      0845
         = $independent: *locstr* _ "Independent";                       0846
         = $user: *locstr* _ "User";                                     0848
         = $server: *locstr* _ "Server";                                 0849
         = $tip: *locstr* _ "Tip";                                       0847
         = $associate: *locstr* _ "Associate";                           0850
         ENDCASE err($"Illegal Organization type");                      0851
      FIND SF(*locstr*) ^value SE(*locstr*) ^t1;                         0852
```

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

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

PSPROGS

```
< NLS, PSPROGS.NLS;17, >, 24-MAR-77 08:07 KJM ;;;;
FILE psprogs % L10 <rel-nls>psprogs %% (110,) (rel-nls,psprogs.rel,) %
                                                                        02
% DECLARATIONS %                                                       03
   DEFINE                                                               04
   prsavesize = 30#,     % enough for 10 replace parse rules %         05
   prrecsize = 3#,       % wordsize of prrecord %                      06
   sbstacksize = 20#,    % size of sbstack %                           07
   sbentsize =2#;        % size of sbentry records %                   08
% PROGRAMS SUBSYSTEM %                                                 09
   (xpattach)            % attach subsystem to NLS %                  0685
      PROCEDURE( resultptr, parsemode, subsysptr,dispptr);             0686
      REF resultptr, subsysptr, dispptr;                               0687
      % --------------- %                                             0688
      CASE parsemode OF                                                0689
         = parsing:                                                    0690
            BEGIN                                                      0691
            % define the subsystem given the specified handle %       0692
               dfnsubsys( dispptr, gtctrlbits(dispptr, $nlssubs),
               $nlssubs);                                              0693
            END;                                                       0694
         ENDCASE;                                                      0695
      RETURN( &resultptr );                                            0696
      END.
                                                                       0697
   (xpcompile) PROCEDURE        % compile program %                   0719
      %FORMALS%                                                        0720
         (resultptr,     %result record pointer%                       0721
         parsemode,      %parsing mode%                                0722
         entity,         %entity command word%                         0723
         location        %selected location to begin%,                 0724
         compiler,       %name of compiler%                            0725
         fname);         %nam of output file%                          0726
      LOCAL                                                            0727
         adstr[40],      % data structure for link parsing %            0728
         savecsp,        % save location for current csp %              0729
         errors,         % error count %                                0730
         tptr2, % local ptr to text ptr %                              0731
         proc,     % name of Procedure or L10 core routine %           0732
         da;             % ptr to display area %                        0733
      LOCAL STRING                                                     0734
         errstr[50],     % error string %                               0735
         rfilname[200]; % output rel-file name %                       0736
      REF resultptr, da, entity, location, compiler, fname, tptr2, proc;
                                                                       0737
      CASE parsemode OF                                                0738
         = parsing:                                                    0739
            BEGIN                                                      0740
            &da _ lda();                                               0741
            CASE entity OF                                             0742
               = 15, =150, =151: % compile file, L10, or Procedure %
                                                                       0743
                  BEGIN                                                0744
                  % set so ^O won't clear output buffer %              0745
                     rubnocob _ TRUE;                                  0746
                  % for displays, clear so user can see errors %       0747
```

```
                IF nlmode=fulldisplay AND defttysim THEN shutdis();
                                                                    0748
             % save dacsp %                                         0749
                savecsp _ da.dacsp;                                 0750
             % call compiler, etc. %                               0751
                IF entity=15 THEN                                   0752
                   BEGIN                                            0753
                   % put file names into local strings %            0754
                      lnkprs( &compiler, $adstr);                   0755
                      CASE lnbfls( &fname, 0, $rfilname) OF         0756
                         = lhostn: NULL;                            0757
                         ENDCASE                                    0758
                            err($"Remote File Manipulations Not
                            Implemented Yet");                      0759
                   da.dacsp _ location;                             0760
                   errors _ cpcmpfl( FALSE, $adstr, $rfilname, &da)
                                                                    0761
                   END                                              0762
                ELSE                                                0763
                   BEGIN %L10 or Procedure%                         0764
                   &proc _ (IF entity=150 THEN $gpl10 ELSE
                   $cpcmproc);                                      0765
                   errors_proc(location, &da);                      0766
                   END;                                             0767
             %restore csp %                                         0768
                da.dacsp _ savecsp;                                 0769
             % tell the user about any errors %                    0770
                IF errors > 0 THEN                                  0771
                   BEGIN                                            0772
                   *errstr* _ STRING(errors), " error(s):  Type
                   CA.";                                            0773
                   dismes (1, $errstr);                             0774
                   clrbuf (0);   % clear the input buffer %         0775
                   LOOP IF inpcuc() = CA THEN EXIT;                 0776
                   dismes (0);                                      0777
                   END;                                             0778
             % recreate display if needed%                         0779
                ckdvln(); %in case we have reconnected%             0780
                continue_TRUE;                                      0781
                initdis();                                          0782
                continue_FALSE;                                     0783
                dpset(dspallf, endfil, endfil, endfil);             0784
             END;                                                   0785
          = 110 %- content -%:  % analyzer filter %                 0786
             BEGIN                                                  0787
             da.dacacode _ cpconan(&location, &da);                 0788
             da.davspec.vscapf _ TRUE;                              0789
             END;                                                   0790
       ENDCASE err(notyet);                                         0791
       END;                                                         0792
ENDCASE;                                                            0793
RETURN( &resultptr );                                               0794
END.
                                                                    0795
(xpdelete)          % Delete a user program from stack %              093
   PROCEDURE (resultptr, parsemode, howmany );                       094
```

```
   REF resultptr, howmany;                                              095
   CASE parsemode OF                                                    096
      = parsing:                                                        097
         CASE howmany OF                                                098
            = 95 %- all -%:  % all programs in the stack %              099
               BEGIN                                                   0711
               gpgmrst();                                              0100
               IF upgbsz > $upgbdf THEN gpbsz($upgbdf);                0714
               END;                                                    0713
            = 79 %- last -%:  % only the last program in the stack %
                                                                       0101
               gppop();                                                0102
         ENDCASE err(notyet);                                          0103
   ENDCASE;                                                            0104
   RETURN (&resultptr);                                                0105
   END.
                                                                       0106
(xpdeinstitute)      % Deinstitute a user program %                    0107
   PROCEDURE (resultptr, parsemode, type);                             0108
   LOCAL t2ptr, da, field;                                             0109
   LOCAL STRING nmstr[50];                                             0110
   REF resultptr, type, t2ptr, da;                                     0111
   CASE parsemode OF                                                   0112
      = parsing:                                                       0113
         BEGIN                                                         0114
         % decode type to determine field %                           0115
            field _                                                    0116
               CASE type OF                                            0117
                  = 110 %- content -%:       % content analyzer
                  program %                                           0118
                     dacacode;                                        0119
                  = 152 %- seqgenerator -%:  % sequence generator
                  program %                                           0120
                     dausqcod;                                        0121
                  = 92 %- sort -%:           % sort key program %
                                                                      0122
                     daukeycod;                                       0123
               ENDCASE err(notyet);                                   0124
         % deinstitute program from current display area %           0125
            &da _ lda();                                             0126
            da.field _ 0;                                            0127
         % make sure we dont undo this in cmdfinish %                0543
            cspupdate _ FALSE;                                       0544
         END;                                                        0128
   ENDCASE;                                                          0129
   RETURN (&resultptr);                                              0130
   END.
                                                                     0131
(xpdetach)           % detach subsystem to NLS %                     0698
   PROCEDURE( resultptr, parsemode, subsysptr,dispptr);              0699
   REF resultptr, subsysptr, dispptr;                                0700
   % --------------- %                                              0701
   CASE parsemode OF                                                 0702
      = parsing:                                                     0703
         BEGIN                                                       0704
         % detach the subsystem given the specified handle %         0705
```

```
            delsubsys( dispptr, $nlssubs);                          0706
         END;                                                       0707
      ENDCASE;                                                      0708
   RETURN( &resultptr );                                            0709
   END.
                                                                    0710

(xpinstitute)        % Institute a user program %                   0156
   PROCEDURE (resultptr, parsemode, pgmnam, type);                  0157
   LOCAL t2ptr, field, da;                                          0158
   LOCAL STRING nmstr[50];                                          0159
   REF resultptr, pgmnam, t2ptr, type, da;                          0160
   CASE parsemode OF                                                0161
      = parsing:                                                    0162
         BEGIN                                                      0163
         % move program name into local string %                    0164
            &t2ptr _ &pgmnam + d2sel;                               0165
            *nmstr* _ pgmnam t2ptr;                                 0166
         % decode type to determine field %                        0167
            field _                                                 0168
               CASE type OF                                         0169
                  = 110 %- content -%:       % content analyzer
                  program %                                         0170
                     dacacode;                                      0171
                  = 152 %- seggenerator -%:  % sequence generator
                  program %                                         0172
                     dausgcod;                                      0173
                  = 92 %- sort -%:           % sort key program %
                                                                    0174
                     daukeycod;                                     0175
               ENDCASE err(notyet);                                 0176
         % institute program into current display area %           0177
            &da _ lda();                                            0178
            da.field _ upgcnv( $nmstr );                            0179
         % make sure we dont undo this in cmdfinish %              0541
            cspupdate _ FALSE;                                      0542
         END;                                                       0180
   ENDCASE;                                                         0181
   RETURN (&resultptr);                                             0182
   END.
                                                                    0183

(xpkill)             % kill tenex subsystem %                       0184
   PROCEDURE (resultptr, parsemode);                                0185
   REF resultptr;                                                   0186
   CASE parsemode OF                                                0187
      = parsing:                                                    0188
         BEGIN                                                      0189
         % kill the tenex subsystem %                               0190
            gpkill();                                               0191
         END;                                                       0192
      ENDCASE;                                                      0193
   RETURN (&resultptr);                                             0194
   END.
                                                                    0195
   %  %                                                            0196
```

```
(xpload)              % load a user program %                          0652
   PROCEDURE (resultptr, parsemode, filnm);                           0653
   LOCAL adstr[40];                                                   0654
   REF resultptr, filnm;                                              0655
   cspupdate _ FALSE;                                                 0656
   CASE parsemode OF                                                  0657
      = parsing:                                                      0658
         BEGIN                                                        0659
         % parse file name link %                                     0660
            lnkprs( &filnm, $adstr);                                  0661
         % load the program into the user programs buffer %           0662
            gpget( $adstr, TRUE % display messages in gpget % );      0663
         END;                                                         0664
   ENDCASE;                                                           0665
   RETURN (&resultptr);                                               0666
   END.
                                                                      0667
(xprun)               % Run a user program %                          0221
   PROCEDURE (resultptr, parsemode, pgmnam);                          0222
   LOCAL t2ptr;                                                       0223
   LOCAL STRING prog[50];                                             0224
   REF resultptr, pgmnam, t2ptr;                                      0225
   CASE parsemode OF                                                  0226
      = parsing:                                                      0227
         BEGIN                                                        0228
         % move filename to local string %                            0229
            &t2ptr _ &pgmnam + d2sel;                                 0230
            *prog* _ pgmnam t2ptr;                                    0231
         % call the program %                                         0232
            gpexpgm( $prog );                                         0233
         END;                                                         0234
   ENDCASE;                                                           0235
   RETURN (&resultptr);                                               0236
   END.
                                                                      0237
%  Commented out (never worked anyway); percents doubled              0796
   (xpreplace)                %% Replace Parserule command %%         0238
      PROCEDURE (resultptr, parsemode, oldrule, filenam, newrule);
                                                                      0239
      LOCAL tp2; REF tp2;                                             0240
      LOCAL instloc, ptr, newloc;  REF instloc, ptr, newloc;          0241
      LOCAL STRING locstr[200];                                       0244
      LOCAL STRING errstr[75];                                        0245
      REF resultptr, oldrule, filenam, newrule;                       0246
      %% ---------------- %%                                         0247
      CASE parsemode OF                                               0248
         = parsing:                                                   0249
            BEGIN                                                     0250
            %% mark the NDDT symbol table to use symbols in the named
            file as locals %%                                         0251
               resultptr _ FALSE; %% in case we have an error %%      0252
               %% move file name to local string %%                  0253
                  CASE lnbfls( &filenam, 0, $locstr) OF               0528
                     = lhostn: NULL;                                  0529
                     ENDCASE                                          0530
                        err($"Remote File Manipulations Not
```

```
                    Implemented Yet");                                  0531
            ddtmark( $locstr );                                         0263
            resultptr _ 1; %% so we will pop the symbol table
            again %%                                                    0264
         %% look up the oldrule %%                                     0265
            &tp2 _ &oldrule + d2sel;                                    0266
            *locstr* _ oldrule tp2;                                     0267
            IF NOT ddtlookup($locstr, FALSE: &instloc) THEN             0268
               BEGIN                                                    0269
               (errout):                                                0270
               *errstr* _ *locstr*, " could not be found";              0271
               err( $errstr );                                          0272
               END;                                                     0273
         %% reset the symbol table mark %%                             0274
            resultptr _ FALSE;                                          0275
            ddtpop();                                                   0276
         %% lookup the new rule %%                                     0277
            &tp2 _ &newrule + d2sel;                                    0278
            *locstr* _ newrule tp2;                                     0279
            IF NOT ddtlookup($locstr, FALSE: &newloc) THEN              0280
               GOTO errout;                                             0281
         %% set ptr to point to a slot in the prsavearea %%            0282
            FOR &ptr _ $prsavearea UP prrecsize UNTIL >=
            $prsavearea+prsavx DO                                       0283
               IF NOT ptr.prexists THEN GOTO buildentry;                0284
         %% allocate a new entry at the end of the stack if there
         is room %%                                                     0285
            IF prsavx + prrecsize > prsavesize                          0286
               THEN err($"Too many parse rules replaced, must
               reset some first");                                      0287
            &ptr _ $prsavearea + prsavx;                                0288
            prsavx _ prsavx + prrecsize;                                0289
         %% initialize a new entry %%                                  0290
            (buildentry):                                               0291
            ptr.prwrd1 _ instloc;                                       0292
            ptr.prwrd2 _ instloc[1];                                    0293
            ptr.praddr _ &instloc;                                      0294
            ptr.prexists _ TRUE;                                        0295
         %% replace the first instruction by an "EXECUTE newrule"
         instruction %%                                                 0296
            instloc.nsuccessor _ 0;                                     0297
            instloc.opcode _ 22; %%*** EXECUTE ***%%                    0298
            instloc.addr _ &newloc;                                     0299
         END;                                                           0300
      = backup, = cleanup:                                              0301
         IF resultptr THEN ddtpop();                                    0302
   ENDCASE;                                                             0303
   RETURN (&resultptr);                                                 0304
   END.
                                                                        0305
%
                                                                        0797
(xpreset)            % Reset command %                                  0306
   PROCEDURE (resultptr, parsemode, entity, rulename, fname);           0307
   LOCAL instptr, ptr, tp2;                                             0308
   REF resultptr, entity, rulename, fname, instptr, ptr, tp2;           0311
```

```
LOCAL STRING locstr[200], errstr[50];                                  0312
% ----------------- %                                                  0313
CASE parsemode OF                                                      0314
   = parsing:                                                          0315
      BEGIN                                                            0316
      resultptr _ FALSE;                                               0317
      CASE entity OF                                                   0318
         = 153 %- buffer -%: % set buffer size %                       0319
          gpbsz( $upgbdf ); % set buffer size to upgbdf pages
          (default) %                                                  0320
         = 154 %- nddt -%:  % NDDT control-h %                         0321
          nddtdisarm();                                                0322
         % commented out (never worked anyway); percents doubled
                                                                       0798
          = 155 %%- parserule -%%:                                     0323
            BEGIN                                                      0324
            %% mark the NDDT symbol table to use symbols in the
            named file as locals %%                                    0325
               resultptr _ FALSE; %% in case we have an error
               %%                                                      0326
               %% move file name to local string %%                   0327
                  CASE lnbfls( &fname, 0, $locstr) OF                  0524
                     = lhostn: NULL;                                   0525
                     ENDCASE                                           0526
                        err($"Remote File Manipulations Not
                        Implemented Yet");                             0527
               ddtmark( $locstr );                                     0337
               resultptr _ 1; %% so we will pop the symbol
               table again %%                                          0338
            %% look up the rulename %%                                0339
               &tp2 _ &rulename + d2sel;                               0340
               *locstr* _ rulename tp2;                                0341
               IF NOT ddtlookup($locstr, FALSE: &instptr) THEN
                                                                       0342
                  BEGIN                                                0343
                  *errstr* _ *locstr*, " could not be found";
                                                                       0344
                  err( $errstr );                                      0345
                  END;                                                 0346
            %% reset the symbol table mark %%                         0347
               resultptr _ FALSE;                                      0348
               ddtpop();                                               0349
            %% find the entry in the prsavearea corresponding
            to the rule name  %%                                       0350
               FOR &ptr _ $prsavearea UP prrecsize UNTIL >=
               $prsavearea+prsavx DO                                   0351
                  IF ptr.praddr THEN                                   0352
                     BEGIN                                             0353
                     %% reset the instruction %%                       0354
                        instptr _ ptr.prwrd1;                          0355
                        instptr[1] _ ptr.prwrd2;                       0356
                        ptr.prexists _ FALSE;                          0357
                        RETURN( &resultptr );                          0358
                     END;                                              0359
            %% didn't find the named rule in the prsavearea,
            tell the user %%                                           0360
```

```
                    err( $"named rule not in the replace stack" );          0361
                                                                            0362
                  END;                                                      0799
            %                                                               0363
            ENDCASE err(notyet);                                            0364
         END;                                                               0365
      = backup, = cleanup:                                                  0366
         IF resultptr THEN ddtpop();                                        0367
   ENDCASE;                                                                 0368
   RETURN (&resultptr);
   END.                                                                     0369

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

```
        END;                                                    0647
      ENDCASE;                                                  0648
RETURN (&resultptr);                                            0649
END.
                                                                0650
%  %                                                            0651
```

```
(xpset)                 % Set command %                                  0438
   PROCEDURE (resultptr, parsemode, entity, param);                     0439
   LOCAL tptr2, i, size;                                                0440
   LOCAL STRING sizestr[20];                                            0441
   REF resultptr, entity, param, tptr2;                                 0442
   CASE parsemode OF                                                    0443
      = parsing:                                                        0444
         CASE entity OF                                                 0445
            = 153 %- buffer -%: % set buffer size %                     0446
               BEGIN                                                    0447
               % move size string into local string, convert to a
               numeric value %                                          0448
                  &tptr2 _ &param + d2sel;                              0449
                  *sizestr* _ param tptr2;                              0450
                  size _ VALUE( $sizestr );                             0451
               % set the buffer to the new size %                       0452
                  IF NOT gpbsz( size ) THEN                             0453
                     err($"Invalid size for programs buffer");          0454
               END;                                                     0455
            = 154 %- nddt -%:  % arm NDDT control-h %                   0456
               BEGIN                                                    0457
               nddtarm();                                               0458
               END;                                                     0459
         ENDCASE err(notyet);                                           0460
   ENDCASE;                                                             0461
   RETURN (&resultptr);                                                 0462
   END.                                                                 0463

(xpsetf)                %set up globals for alternate directory in lsel%
                                                                        0668
   PROCEDURE (resultptr, parsemode);                                    0669
   REF resultptr;                                                       0670
   CASE parsemode OF                                                    0671
      = parsing:                                                        0672
         BEGIN                                                          0673
         %set flag indicating presence of alteratte directory%          0674
            altdfl _ TRUE;                                              0675
         %set up alternate directory and alternate extension strings%
                                                                        0676
            *altdir* _ "PROGRAMS,"; %to add more separate by commas%
                                                                        0677
            *altext* _ "REL,CA,SK,SG,SUBSYS,CML,PROC-REP,";             0678
         END;                                                           0679
      = cleanup: %reset the global flag%                                0680
         altdfl _ FALSE;                                                0681
   ENDCASE;                                                             0682
   RETURN (&resultptr);                                                 0683
   END.                                                                 0684

(xpshow)                % Show Status of user programs display %         0464
   PROCEDURE (resultptr, parsemode);                                    0465
   LOCAL da;                                                            0466
   LOCAL STRING statusstr[400];                                         0467
   REF resultptr, da;                                                   0468
   CASE parsemode OF                                                    0469
      = parsing:                                                        0470
```

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

PSSENDMAIL

```
< NLS, PSSENDMAIL.NLS;24, >, 7-JUN-77 10:00 JDH ;;;;
** This file also exists in <nine, sendmail>.  Be sure to copy any
relevant changes to that file! **                                    0936
FILE pssendmail % L10 to <rel-nls>pssendmail.rel % % (110.sav,)
(rel-nls, pssendmail.rel,) %                                          02
DECLARE EXTERNAL STRING %for send mail status and forms%             0725
   sjnumber= "NUMBER:",                                              0726
   sjtitle= "TITLE:",                                                0727
   sjcomment= "COMMENT:",                                            0730
   sjauthor= "AUTHOR(S):",                                           0731
   sjaction= "DISTRIBUTE FOR ACTION TO:",                            0732
   sjinfo= "DISTRIBUTE FOR INFO-ONLY TO:",                           0733
   sjsubcol= "SUBCOLLECTION(S):",                                    0728
   sjkeywords= "KEYWORD(S):",                                        0729
   sjhandling= "HANDLING INSTRUCTION:",                              0734
   sjrecording= "RECORDING INSTRUCTION:",                            0736
   sjhardcopy= "OFFLINE ITEM -- LOCATED AT:",                        0737
   sjrfc= "RFC NUMBER:",                                             0738
   sjobsoletes= "OBSOLETES ITEM NUMBER(S):",                         0735
   sjaccess= "ACCESS STATUS:",                                       0739
   sjupdates= "UPDATE TO ITEM NUMBER(S):",                           0740
   sjlink= "INSERT LINK TO FOLLOW:",                                 0741
   sjforward= "FORWARD ITEM NUMBER:",                                0742
   sjmessage= "MESSAGE:",                                            0749
   sjbranch= "BRANCH AT:",                                           0743
   sjplex= "PLEX AT:",                                               0744
   sjgroup= "GROUP AT:",                                             0745
   sjfile= "FILE:",                                                  0746
   sjsendit= "SEND THE MAIL.",                                       0747
   sjbacksendit= ".LIAM EHT DNES";                                   0748
% SENDMAIL SUBSYTEM %
                                                                      03
   (xjzapworfil)        % initialize work file %                     012
      PROCEDURE (resultptr, parsemode);                              013
      REF resultptr;                                                 014
      LOCAL STRING locstr[200];                                      015
      CASE parsemode OF                                              016
         = parsing:                                                  017
            BEGIN                                                    018
            IF jworkstid THEN resetf(jworkstid.stfile)               019
            ELSE                                                     020
               BEGIN                                                 021
               *locstr* _ "$JWORK-", *initsr*, "$.SYSTEM;";          022
               jworkstid _ openwk(0, $locstr);                       023
               END;                                                  024
            FIND SF(jworkstid) ^jwp1;                                025
            initjwork();                                            0255
            END;                                                     026
      ENDCASE;                                                       027
      RETURN( &resultptr );                                          028
      END.
                                                                     029
   (xjloaworfil)        % load Journal work file %
   PROCEDURE (resultptr, parsemode);  %returns TRUE if old file being
   used, FALSE if new file created and initialized%                  030
      REF resultptr;                                                 032
```

```
   LOCAL STRING locstr[200];                                          033
   CASE parsemode OF                                                  034
      = parsing:                                                      035
         BEGIN                                                       0718
         IF NOT jworkstid.stfile THEN                                0268
            BEGIN                                                    036
            *locstr* _ '<,  *userstr*, '>, "$SEND-MAIL-", *initsr*,
            "$.NLS", fvrdchar, "1;P707000";                          037
            ON SIGNAL ELSE                                           038
               BEGIN %must create it%                                039
               ON SIGNAL ELSE;                                       040
               jworkstid _ openwk(0, $locstr);                       041
               GOTO ok;                                             0262
               END;                                                  044
            jworkstid _ orgstid;                                     045
            jworkstid.stfile _ open(0, $locstr);                     046
            (ok):  %finishup and leave%                              0263
            FIND SF(jworkstid) ^jwp1;                                047
            [flntadr(jworkstid.stfile)].flnoclos _ TRUE;             0264
            END;                                                     048
         initjwork()                                                 0716
         END;                                                        0717
   ENDCASE;                                                          049
   RETURN( &resultptr );                                             050
   END.

                                                                     051
% not called                                                        0870
(xjcloworfil)        %% close work file %%                          052
   PROCEDURE (resultptr, parsemode);                                 053
   REF resultptr;                                                    054
   CASE parsemode OF                                                 055
      = parsing:                                                     056
         BEGIN                                                       057
         IF jworkstid.stfile THEN                                   0266
            [flntadr(jworkstid.stfile)].flnoclos _ FALSE;           0267
         jworkstid _ 0;                                              059
         END;                                                        060
      ENDCASE;                                                       061
   RETURN( &resultptr );                                             062
   END.

                                                                     063
%                                                                    0871
(xjreserve)          % reserve numbers %                            0878
   PROCEDURE (resultptr, parsemode, type, numb, insertloc);          0879
   LOCAL TEXT POINTER z1, z2, z3;                                    0880
   LOCAL str, typsad, reconnerr;  REF str;                           0881
   REF resultptr, type, numb, insertloc;                             0882
   CASE parsemode OF                                                 0883
      = parsing:                                                     0884
         BEGIN                                                       0885
         IF insertloc THEN dpset (dsprfmt, insertloc, endfil,
         insertloc);                                                 0910
         typsad _ CASE type OF                                       0886
            =103: $"JOURNAL";                                        0887
            ENDCASE &type;                                           0888
         IF NOT &str _ getstring(200, $dspblk) THEN                  0889
```

```
            err($"Storage allocator out of space");                    0890
         gcatnums (&str, $initsr, typsad, IF NOT numb THEN 1 ELSE
         getpint(&numb, &numb+d2sel), 1 : reconnerr);                  0891
         IF reconnerr THEN dismes(2, sysmsg);                          0892
            %warn user he didn't get reconnected%                      0893
         FIND SF(*str*) ^z1 SE(*str*) < CH > ^z2;                      0894
         IF insertloc THEN %insertloc it as a visible%                 0895
            BEGIN                                                      0896
            z3 _ insertloc[d2sel];  z3[1] _ insertloc[d2sel+1];        0911
            cinstex ($z3, $z1, $z2, TRUE);                             0897
            END                                                        0898
         ELSE %just show him%                                          0899
            fbctl (typecalit, &str);                                   0900
         resultptr _ z1; resultptr[1] _ z1[1];                         0901
         resultptr[d2sel] _ z2; resultptr[d2sel+1] _ z2[1];            0902
         resultptr[4] _ &str;                                          0903
         END;                                                          0904
      = backup, = cleanup: IF resultptr[4] THEN                        0905
         freestring(resultptr[4] := 0, $dspblk);                       0906
      ENDCASE;                                                         0907
   RETURN( &resultptr );                                               0908
   END.
                                                                       0909
(xjrfcreserve)       % reserve an RFC and a Catalog numstr %
PROCEDURE (resultptr, parsemode, titlestr, authstr, sendto,
onlineflag, insertloc);                                                072
   LOCAL TEXT POINTER z1, z2;                                          0406
   LOCAL STRING                                                        0407
      auth[50], titl[200], send[150], rfcnum[50], numstr[100];         0411
   REF resultptr, titlestr, authstr, sendto, onlineflag, insertloc;
                                                                       074
   CASE parsemode OF                                                   075
      = parsing:                                                       076
         BEGIN                                                         0398
         %build strings for rfcex%                                     0423
            z2 _ authstr[d2sel]; z2[1] _ authstr[d2sel+1];             0408
            *auth* _ authstr z2;                                       0415
               IF NOT auth.L THEN *auth* _ *initsr*;                   0606
            z2 _ sendto[d2sel]; z2[1] _ sendto[d2sel+1];               0416
            *send* _ sendto z2;                                        0417
            z2 _ titlestr[d2sel]; z2[1] _ titlestr[d2sel+1];           0412
            *titl* _ titlestr z2;                                      0418
         rfcex($auth, $titl, $send, onlineflag, $rfcnum, $numstr);
                                                                       0400
         *numstr* _ "RFC numstr: ", *rfcnum*, ", Catalog numstr: ",
         *numstr*;                                                     0405
         IF insertloc THEN %insertloc it as a visible%                 0402
            BEGIN                                                      0419
            FIND SF(*numstr*) ^z1 SE(*numstr*) ^z2;                    0420
            cinstex (&insertloc, $z1, $z2, TRUE);                      0421
            END                                                        0422
         ELSE %just show him%                                          0403
            fbctl (typecalit, $numstr);                                0401
         END;                                                          0399
   ENDCASE;                                                            077
   RETURN( &resultptr );                                               078
```

```
   END.
                                                                        079
(xjrfcsho)              % show status of RFC reserve requestr %
PROCEDURE (resultptr, parsemode, titlestr, authstr, sendto,
onlineflag);                                                            0432
   LOCAL TEXT POINTER aptr, tptr, dptr;                                 0433
   LOCAL STRING str[200];                                               0434
   REF resultptr, titlestr, authstr, sendto, onlineflag;                0436
   CASE parsemode OF                                                    0437
      = parsing:                                                        0438
         BEGIN                                                          0439
         %build text-pointers for concatinating status message%         0440
            aptr _ authstr[d2sel]; aptr[1] _ authstr[d2sel+1];          0441
            dptr _ sendto[d2sel]; dptr[1] _ sendto[d2sel+1];            0443
            tptr _ titlestr[d2sel]; tptr[1] _ titlestr[d2sel+1];        0445
         *str* _                                                        0448
            "Title: ", titlestr tptr, "
            Author(s): ", authstr aptr, "
            Send to: ", sendto dptr;                                    0460
         IF onlineflag THEN                                             0461
            *str* _ *str*, "
            Online document"                                            0462
         ELSE                                                           0463
            *str* _ *str*, "
            Offline document";                                          0464
         %show him%                                                     0454
            fbctl (typelit, $str);                                      0455
         END;                                                           0456
   ENDCASE;                                                             0457
   RETURN( &resultptr );                                                0458
   END.
                                                                        0459
(xjdoit)                % finish submission or forward request %         088
   PROCEDURE (resultptr, parsemode);                                     089
   REF resultptr;                                                        090
   CASE parsemode OF                                                     093
      = parsing:                                                         094
         BEGIN                                                           095
         jsubmit();                                                      096
         END;                                                            097
   ENDCASE;                                                              098
   RETURN( &resultptr );                                                 099
                                                                        0100
   END.
                                                                        0101
(xjlock)                % lock or unlock the journal %                  0689
   PROCEDURE (resultptr, parsemode, password, lockflag);                0690
   LOCAL TEXT POINTER z1, z2;                                           0691
   LOCAL STRING passstr[20];                                            0692
   REF resultptr, password, lockflag;                                   0693
   CASE parsemode OF                                                    0694
      = parsing:                                                        0695
         BEGIN                                                          0696
         % get information out of the parameter records %               0697
            z1 _ password;                                              0698
            z1[1] _ [&password + 1];                                    0699
```

```
            z2 _ [&password + 2];                                          0700
            z2[1] _ [&password + 3];                                       0701
            *passstr* _ z1 z2;                                             0702
         % Check the password value. %                                     0703
            IF *passstr* # *jnlpsw* THEN                                   0704
               BEGIN                                                       0705
               dismes(1, $"Illegal password");                             0706
               EXIT CASE;                                                  0707
               END;                                                        0708
         % Lock or unlock the journal %                                    0709
            IF lockflag THEN lockjo(0 %jlock flag%)                        0710
            ELSE unlkjo(0);                                                0711
         END;                                                              0712
      ENDCASE;                                                             0713
   RETURN( &resultptr );                                                   0714
   END.
                                                                           0715
% not called                                                              0872
(xjpriharcop)          %% print hard copy %%                              0110
   PROCEDURE (resultptr, parsemode);                                       0111
   REF resultptr;                                                          0112
   CASE parsemode OF                                                       0113
      = parsing: err(notyet);                                              0114
   ENDCASE;                                                                0115
   RETURN( &resultptr );                                                   0116
   END.
                                                                           0117
%                                                                          0873
(xjprocess)            % process command form %                           0118
   PROCEDURE (resultptr, parsemode, destination);                          0119
   REF resultptr, destination;                                             0120
   LOCAL TEXT POINTER left, right, z1, z2;                                 0284
   LOCAL STRING locstr[1500];                                              0379
   CASE parsemode OF                                                       0121
      = parsing:                                                           0122
         BEGIN                                                             0343
         %Number%                                                          0372
            IF FIND SF(destination) [*sjnumber*] $SP ^left [EOL] < CH
            $SP ^right > THEN                                              0373
               BEGIN                                                       0377
               *locstr* _ left right;                                      0375
               IF locstr.L THEN setjnumber ($locstr)                       0376
               ELSE %assign number and update form%                        0380
                  BEGIN                                                    0381
                  gcatnums ($locstr, $initsr, $"JOURNAL", 1, 1);           0505
                  IF locstr.L THEN                                         0506
                     BEGIN                                                 0511
                     setjnumber ($locstr);                                 0509
                     ST left right _ *locstr*;                             0382
                     END;                                                  0510
                  END;                                                     0383
               END;                                                        0378
         %title%                                                           0289
            IF FIND SF(destination) [*sjtitle*] $SP ^left [EOL] < CH
            $SP ^right > THEN                                              0345
               BEGIN                                                       0508
```

```
        *locstr* _ left right;                                  0290
        IF locstr.L THEN setjtitle ($locstr);                   0291
        END;                                                    0507
    %comment%                                                   0347
       IF FIND SF(destination) > [*sjcomment*] $SP ^left [EOL] <
       CH $SP ^right > THEN                                     0348
        BEGIN                                                   0512
        *locstr* _ left right;                                  0513
        IF locstr.L THEN setjcomment ($locstr);                 0514
        END;                                                    0515
    %author%                                                    0352
       IF FIND SF(destination) > [*sjauthor*] $SP ^left [EOL] <
       CH $SP ^right > THEN                                     0353
        BEGIN                                                   0516
        *locstr* _ left right; %supposedly an ident string%
                                                                0517
        IF locstr.L THEN setjauthor ($locstr);                  0518
        END;                                                    0519
    %Action distribution%                                       0357
       IF FIND SF(destination) > [*sjaction*] $SP ^left [EOL] <
       CH $SP ^right > THEN                                     0358
        BEGIN                                                   0520
        *locstr* _ left right; %supposedly an ident string%
                                                                0524
        IF locstr.L THEN setjaction ($locstr);                  0525
        END;                                                    0523
    %Information-only distribution%                             0362
       IF FIND SF(destination) > [*sjinfo*] $SP ^left [EOL] < CH
       $SP ^right > THEN                                        0363
        BEGIN                                                   0526
        *locstr* _ left right; %supposedly an ident string%
                                                                0530
        IF locstr.L THEN setjinfo ($locstr);                    0531
        END;                                                    0529
    %subcollections%                                            0367
       IF FIND SF(destination) > [*sjsubcol*] $SP ^left [EOL] <
       CH $SP ^right > THEN                                     0368
        BEGIN                                                   0532
        *locstr* _ left right; %supposedly an ident string%
                                                                0536
        IF locstr.L THEN setjsubcol ($locstr);                  0537
        END;                                                    0535
    %keywords%                                                  0465
       IF FIND SF(destination) > [*sjkeywords*] $SP ^left [EOL]
       < CH $SP ^right > THEN                                   0538
        BEGIN                                                   0539
        *locstr* _ left right;                                  0540
        IF locstr.L THEN setjkeyw ($locstr);                    0541
        END;                                                    0542
    %handling instructions%                                     0470
       IF FIND SF(destination) > [*sjhandling*] $SP ^left [EOL]
       < CH $SP ^right > THEN                                   0543
        BEGIN                                                   0544
        *locstr* _ left right;                                  0545
        IF locstr.L THEN setjexpedite (*locstr* = "Expedite");
                                                                0546
```

```
      END;                                                          0547
%recording instructions%                                            0500
   IF FIND SF(destination) > [*sjrecording*] $SP ^left [EOL]
   < CH $SP ^right > THEN                                           0548
      BEGIN                                                         0549
      *locstr* _ left right;                                        0550
      IF locstr.L THEN                                              0551
         setjexpedite (*locstr* = "Unrecorded");                    0553
      END;                                                          0552
%hardcopy location%                                                 0475
   IF FIND SF(destination) > [*sjhardcopy*] $SP ^left [EOL]
   < CH $SP ^right > THEN                                           0554
      BEGIN                                                         0555
      *locstr* _ left right;                                        0556
      IF locstr.L THEN                                              0557
         BEGIN                                                      0561
         FIND SF(*locstr*) ^left SE(*locstr*) ^right;               0559
         setjsource (hcopyv, $left, $right);                        0560
         END;                                                       0562
      END;                                                          0558
%rfc number%                                                        0480
   IF FIND SF(destination) > [*sjrfc*] $SP ^left [EOL] < CH
   $SP ^right > THEN                                                0563
      BEGIN                                                         0564
      *locstr* _ left right;                                        0565
      IF locstr.L THEN setjrfc ($locstr);                           0566
      END;                                                          0571
%obsoletes%                                                         0485
   IF FIND SF(destination) > [*sjobsoletes*] $SP ^left [EOL]
   < CH $SP ^right > THEN                                           0572
      BEGIN                                                         0573
      *locstr* _ left right;                                        0574
      IF locstr.L THEN setjobsoletes ($locstr);                     0575
      END;                                                          0576
%access status%                                                     0678
   IF FIND SF(destination) > [*sjaccess*] $SP ^left [EOL] <
   CH $SP ^right > THEN                                             0679
      BEGIN                                                         0680
      *locstr* _ left right;                                        0681
      chprvsts (jwp1.stfile, (IF *locstr* = "PRIVATE"               0684
         THEN $psprivate                                            0687
         ELSE $pspublic));                                          0688
      END;                                                          0683
%updates%                                                           0490
   IF FIND SF(destination) > [*sjupdates*] $SP ^left [EOL] <
   CH $SP ^right > THEN                                             0577
      BEGIN                                                         0578
      *locstr* _ left right;                                        0579
      IF locstr.L THEN setjupdates ($locstr);                       0580
      END;                                                          0581
%insert link%                                                       0495
   IF FIND SF(destination) > [*sjlink*] $SP ^left [EOL] < CH
   $SP ^right > THEN                                                0582
      BEGIN                                                         0583
      *locstr* _ left right;                                        0584
      IF locstr.L THEN setjlink ($locstr);                          0585
```

```
      END;                                                          0586
%message%                                                           0607
   IF FIND SF(destination) > [*sjmessage*] $SP ^left [EOL] <
   CH $SP ^right > THEN                                             0608
      BEGIN                                                         0609
      IF POS right > left THEN setjsource (textv, $left,
      $right);                                                      0611
      END;                                                          0612
%branch%                                                            0613
   IF FIND SF(destination) > [*sjbranch*] $SP ^left [EOL] <
   CH $SP ^right > THEN                                             0614
      BEGIN                                                         0615
      z1 _ right;  z1[1] _ right[1];                                0860
      caddexp($left, $right, lda(), $z1);                           0618
      IF z1 NOT= endfil THEN setjsource (groupv, $z1, $z1);
                                                                    0619
      END;                                                          0617
%plex%                                                              0626
   IF FIND SF(destination) > [*sjplex*] $SP ^left [EOL] < CH
   $SP ^right > THEN                                                0627
      BEGIN                                                         0628
      z1 _ right;  z1[1] _ right[1];                                0858
      caddexp($left, $right, lda(), $z1);                           0629
      IF z1 NOT= endfil THEN                                        0630
         BEGIN                                                      0642
         left _ gethed(z1);                                         0638
         right _ getail(z1);                                        0639
         left[1] _ right[1] _ 1;                                    0640
         setjsource (groupv, $left, $right);                        0641
         END;                                                       0643
      END;                                                          0631
%group%                                                             0644
   IF FIND SF(destination) > [*sjgroup*] $SP ^left [EOL] <
   CH $SP ^right > THEN                                             0645
      BEGIN                                                         0646
      z1 _ right;  z1[1] _ right[1];                                0859
      caddexp($left, $right, lda(), $z1);                           0647
      IF z1 NOT= endfil THEN                                        0648
         IF FIND right > [EOL] $SP ^left [EOL] < CH $SP
         ^right > THEN                                              0657
            BEGIN                                                   0649
            z2 _ right;  z2[1] _ right[1];                          0953
            caddexp($left, $right, lda(), $z2);                     0658
            z1 _ grptst(z1, z2 : z2);                               0656
            z1[1] _ z2[1] _ 1;                                      0652
            setjsource (groupv, $z1, $z2);                          0653
            END;                                                    0654
      END;                                                          0655
%file%                                                              0632
   IF FIND SF(destination) > [*sjfile*] $SP ^left [EOL] < CH
   $SP ^right > THEN                                                0633
      BEGIN                                                         0634
      IF POS right > left THEN setjsource (filev, $left,
      $right);                                                      0636
      END;                                                          0637
%send it%                                                           0333
```

```
            IF FIND SE(destination) < *sjbacksendit* EOL THEN     0334
               jsubmit();                                         0337
         END;                                                     0344
   ENDCASE;                                                       0123
   RETURN( &resultptr );                                          0124
   END.
                                                                  0125
(xjinsstatus)        % Insert Status of submission form %         0126
   PROCEDURE (resultptr, parsemode, destination, level);          0127
   LOCAL TEXT POINTER z1, z2, z3;                                 0600
   LOCAL STRING string[2000];                                     0261
   REF resultptr, destination, level;                             0128
   CASE parsemode OF                                              0129
      = parsing:                                                  0130
         BEGIN                                                    0257
         jstatus ($string);                                       0258
         cnvcrlfteol($string); %convert string CRLF's to EOL's%   0942
         IF FIND SF(*string*) ^z1 ["FILE: "] ^z2 [EOL] < CH > ^z3
         THEN  %make process sendmail form work for files%        0940
            *string* _ z1 z2, '<, z2 z3, '>, z3 SE(*string*);     0941
         IF FIND SF(*string*) ^z1 ["[THROUGH]"] ^z3 < [ '[ ] > ^z2
         THEN  %make process sendmail form work for groups%       0949
            *string* _ z1 z2, z3 SE(*string*);                    0950
         FIND SF(*string*) ^z1 SE(*string*) ^z2;                  0259
         curmkr _ cinssta(destination, level, $z1, $z2);          0601
         dpset(dspstrc, curmkr, endfil, curmkr);                  0603
         curmkr[1] _ 1;                                           0602
         END;                                                     0260
   ENDCASE;                                                       0131
   RETURN( &resultptr );                                          0132
   END.
                                                                  0133
(xjinsrecord)        % Insert Status of ident record %
PROCEDURE (resultptr, parsemode, identptr, destination, level);   0912
   LOCAL rectype;                                                 0913
   LOCAL TEXT POINTER z1, z2;                                     0914
   LOCAL STRING idstring[500], recstring[2000], string[2000];     0915
   REF resultptr, identptr, destination, level;                   0916
   CASE parsemode OF                                              0917
      = parsing:                                                  0918
         BEGIN                                                    0919
         z1 _ identptr[d2sel]; z1[1] _ identptr[d2sel+1];         0920
         *idstring* _ identptr z1;                                0921
         IF NOT ckident($idstring, $recstring, idfno) THEN        0922
            err($"Illegal Ident");                                0923
         rectype _ IF jgrptst($recstring, 0) THEN grptyp          0924
            ELSE IF orgtst($recstring, 0) THEN orgtyp ELSE indtyp;
                                                                  0925
         jidstatus ($idstring, $recstring, rectype, $string);     0926
         cnvcrlfteol($string); %convert string CRLF's to EOL's%   0943
         FIND SF(*string*) ^z1 SE(*string*) ^z2;                  0927
         curmkr _ cinssta(destination, level, $z1, $z2);          0928
         dpset(dspstrc, curmkr, endfil, curmkr);                  0929
         curmkr[1] _ 1;                                           0930
         END;                                                     0931
      = backup, = cleanup: IF resultptr[4] THEN
```

```
      freestring(resultptr[4] := 0, $dspblk);                          0932
      ENDCASE;                                                         0933
   RETURN( &resultptr );                                               0934
   END.
                                                                       0935
(xjcopycontent)      %Copy content of message%
PROCEDURE (resultptr, parsemode, destination, level);                  0954
   LOCAL stid1, stid2;                                                 0955
   REF resultptr, destination, level;                                  0956
   CASE parsemode OF                                                   0957
      = parsing:                                                       0958
         IF (stid1 _ getnxt(jwp1)) # endfil THEN %there's content%
                                                                       0959
            BEGIN                                                      0960
            stid2 _ getail(stid1);  %a group%                          0961
            stid1 _ ccopgro(destination, level, stid1, stid2, 0, 0);
                                                                       0962
            dpset(dspstrc, stid1, endfil, dpstp(destination));         0963
            END                                                        0964
         ELSE                                                          0965
            err($"No content specified yet.");                         0966
      ENDCASE;                                                         0967
   RETURN(&resultptr);                                                 0968
   END.                                                                0969
(xjstatus)           % Show Status of submission request %             0587
   PROCEDURE (resultptr, parsemode);                                   0588
   LOCAL STRING string[2000];                                          0589
   REF resultptr;                                                      0590
   CASE parsemode OF                                                   0591
      = parsing:                                                       0592
         BEGIN                                                         0593
         jstatus ($string);                                            0594
         fbctl (typecalit, $string);                                   0595
         END;                                                          0596
   ENDCASE;                                                            0597
   RETURN( &resultptr );                                               0598
   END.
                                                                       0599
(xjistatus)  % Show Status of ident record %
PROCEDURE (resultptr, parsemode, identptr, fieldname);                 0750
   LOCAL rectype;                                                      0751
   LOCAL TEXT POINTER z1;                                              0752
   LOCAL STRING idstring[500], recstring[2000], string[2000];          0753
   REF resultptr, identptr, fieldname;                                 0754
   CASE parsemode OF                                                   0755
      = parsing:                                                       0756
         BEGIN                                                         0757
         z1 _ identptr; z1[1] _ identptr[d2sel+1];                     0758
         *idstring* _ identptr z1;                                     0759
         IF NOT ckident($idstring, $recstring, idfno) THEN             0760
            err($"Illegal Ident");                                     0761
         rectype _ IF jgrptst($recstring, 0) THEN grptyp               0762
            ELSE IF orgtst($recstring, 0) THEN orgtyp ELSE indtyp;
                                                                       0763
         jidstatus ( $idstring, $recstring, rectype, $string);         0764
         IF fieldname = 69 %- undelete -% THEN                         0937
```

```
            IF FIND SF(*string*) ["Comments: DELETED"] THEN          0938
               err($"That ident has been deleted");                  0939
            fbctl(typecalit, $string);                               0765
            END;                                                     0766
         = backup, = cleanup: IF resultptr[4] THEN
         freestring(resultptr[4] := 0, $dspblk);                     0767
         ENDCASE;                                                    0768
      RETURN( &resultptr );                                          0769
      END.
                                                                     0770
(xjforward)          % Forward J doc to new ids %                   0134
      PROCEDURE (resultptr, parsemode, jnumber, actionflag, ids);    0135
      REF resultptr, jnumber, actionflag, ids;                       0136
      LOCAL TEXT POINTER z2;                                         0864
      LOCAL STRING numberstr[20], idlst[10000];                      0865
      CASE parsemode OF                                              0137
         = parsing:                                                  0138
            BEGIN                                                    0861
            FIND jnumber > $NP ^jnumber $D ^z2;                      0866
            *numberstr* _ jnumber z2;                                0867
            z2 _ ids[d2sel];  z2[1] _ ids[d2sel+1];                  0868
            *idlst* _ ids z2;                                        0869
            secdist($numberstr, $idlst, (IF actionflag = 137 %+ action
            +% THEN TRUE ELSE FALSE));                               0863
            END;                                                     0862
      ENDCASE;                                                       0139
      RETURN( &resultptr );                                          0140
      END.
                                                                     0141
(jsetfield)          %set the specified field to the specified value%
                                                                     0142
      PROCEDURE (resultptr, parsemode, field, value);                0143
      LOCAL proced;                                                  0144
      LOCAL TEXT POINTER t1, t2;                                     0145
      LOCAL STRING                                                   0146
         locstr[10000];  %for passing strings to core routines%      0147
      REF resultptr, field, value, proced;                           0148
      CASE parsemode OF                                              0149
         = parsing:                                                  0150
            BEGIN                                                    0151
            t1 _ value[d2sel];  t1[1] _ value[d2sel+1];              0152
            CASE field OF                                            0153
               =137 %- action -%:  &proced _ $setjaction;            0154
               =138 %- authors -%:  &proced _ $setjauthor;           0159
               =1 %- branch -%, =3 %- plex -%, =2 %- group -%:       0167
                  BEGIN                                              0168
                  setjsource(groupv, &value, $t1);                   0169
                  RETURN (&resultptr);                               0385
                  END;                                               0170
               =139 %- comment -%:  &proced _ $setjcomment;          0171
               =140 %- expedite -%:                                  0176
                  BEGIN                                              0177
                  setjexpedite(value);                               0178
                  RETURN (&resultptr);                               0386
                  END;                                               0179
               =15 %- file -%: %number%                              0180
```

```
                BEGIN                                                   0181
                filnam(value.stfile, $locstr);                          0659
                FIND SF(*locstr*) ^value SE(*locstr*) ^t1;              0660
                setjsource(filev, &value, $t1);                         0662
                RETURN (&resultptr);                                    0387
                END;                                                    0183
              =141 %- hardcopy -%:                                      0189
                BEGIN                                                   0190
                setjsource(hcopyv, &value, $t1);                        0191
                RETURN (&resultptr);                                    0389
                END;                                                    0193
              =142 %- information -%:  &proced _ $setjinfo;             0194
              =143 %- insert -%: %link%                                 0199
                BEGIN                                                   0200
                setjlink($t1);                                          0201
                RETURN (&resultptr);                                    0390
                END;                                                    0202
              =144 %- keywords -%:  &proced _ $setjkeyword;             0203
              =11 %- number -%:                                         0208
                BEGIN                                                   0720
                *locstr* _ value t1;                                    0724
                FIND SF(*locstr*) ^value $D ^t1;                        0721
                &proced _ $setjnumber;                                  0719
                END;                                                    0723
              =145 %- obsoletes -%:  &proced _ $setjobsoletes;          0213
              =115 %- private -%:                                       0668
                BEGIN                                                   0672
                chprvsts (jwp1.stfile, $psprivate);                     0669
                RETURN (&resultptr);                                    0677
                END;                                                    0675
              =116 %- public -%:                                        0670
                BEGIN                                                   0673
                chprvsts (jwp1.stfile, $pspublic);                      0671
                RETURN (&resultptr);                                    0676
                END;                                                    0674
              =146 %- rfc -%:  &proced _ $setjrfc;                      0218
              =4 %- statement -%:                                       0223
                BEGIN                                                   0224
                setjsource(stmtv, &value, $t1);                         0225
                RETURN (&resultptr);                                    0391
                END;                                                    0226
              =147 %- subcollections -%:  &proced _ $setjsubcol;        0227
              =148 %- title -%:  &proced _ $setjtitle;                  0232
              =12 %- text -%:                                           0944
                BEGIN                                                   0945
                setjsource(textv, &value, $t1);                         0946
                RETURN (&resultptr);                                    0947
                END;                                                    0948
              =149 %- unrecorded -%:                                    0237
                BEGIN                                                   0238
                setjunrecorded (TRUE);                                  0384
                RETURN (&resultptr);                                    0392
                END;                                                    0243
              =53 %- update -%:  &proced _ $setjupdates;                0244
              ENDCASE err(notyet);                                      0249
            %call appropriate setj procedure%                           0163
```

```
            *locstr* _ value t1;                                    0164
            proced( $locstr );                                      0165
         END;                                                       0250
      ENDCASE;                                                      0251
   RETURN( &resultptr );                                            0252
   END.
                                                                    0253
(jidstatus) %generate status string for an ident entry%
PROCEDURE (idstr, entrystr, typid, statstr);                        0771
   %This procedure inserts the status of the entry passed into the
   status string.%                                                  0772
   LOCAL TEXT POINTER ptr1, ptr2, ptr3, ptr4;                       0773
   LOCAL f1, f2;                                                    0774
   LOCAL STRING cadstr[300];                                        0775
   REF idstr, statstr;                                              0776
   *statstr* _ "Ident: ", *idstr*, CR, LF;                          0777
   getinam(entrystr, 0, $ptr1, $ptr2);                              0778
   *statstr* _ *statstr*, "Name: ", ptr1 ptr2, CR, LF;              0779
   IF typid = indtyp THEN                                           0780
      BEGIN                                                         0781
      getiorg(entrystr, 0, $ptr1, $ptr2);                           0782
      *statstr* _ *statstr*, "Organization: ", ptr1 ptr2, CR, LF;
                                                                    0783
      IF getisorg(entrystr, 0, $ptr1, $ptr2) THEN                   0784
         *statstr* _ *statstr*, "Secondary Organization(s): ",      0785
            ptr1 ptr2, CR, LF;                                      0786
      END                                                           0787
   ELSE                                                             0788
      BEGIN                                                         0789
      IF expdtst(entrystr, 0) THEN                                  0790
         *statstr* _ *statstr*, "Expand", CR, LF                    0791
      ELSE *statstr* _ *statstr*, "Unexpanded", CR, LF;             0792
      getimem(entrystr, 0, $ptr1, $ptr2);                           0793
      *statstr* _ *statstr*, "Membership: ", ptr1 ptr2, CR, LF;     0794
      geticord(entrystr, 0, $ptr1, $ptr2);                          0795
      *statstr* _ *statstr*, "Coordinator: ", ptr1 ptr2, CR, LF;    0796
      IF typid = orgtyp THEN                                        0797
         BEGIN                                                      0798
         getityp(entrystr, 0, $ptr1, $ptr2);                        0799
         *statstr* _ *statstr*, "Organization Type: ", ptr1 ptr2, CR,
         LF;                                                        0800
         END;                                                       0801
      END;                                                          0802
   IF getigrps (entrystr, 0, $ptr1, $ptr2) THEN                     0803
      *statstr* _ *statstr*, "Groups: ", ptr1 ptr2, CR, LF;         0804
   %Put Online and Network addresses in cadstr%                     0805
      *cadstr* _ NULL;                                              0806
      f2 _ getinlhost(entrystr, 0, $ptr3, $ptr4);                   0807
      IF (f1 _ getiuser(entrystr, 0, $ptr1, $ptr2)) OR f2 THEN      0808
         BEGIN                                                      0809
         *cadstr* _ *cadstr*, "  Online(NLS)", CR, LF;              0810
         IF f1 THEN *cadstr* _ *cadstr*, "    User: ", ptr1 ptr2, CR,
         LF;                                                        0811
         IF f2 THEN *cadstr* _ *cadstr*, "    Host: ", ptr3 ptr4, CR,
         LF;                                                        0812
         END;                                                       0813
```

```
    f2 _ getihost(entrystr, 0, $ptr3, $ptr4);                          0814
    IF (f1 _ getinma(entrystr, 0, $ptr1, $ptr2)) OR f2 THEN            0815
       BEGIN                                                           0816
       *cadstr* _ *cadstr*, "  Network", CR, LF;                       0817
       IF f1 THEN *cadstr* _ *cadstr*, "    User: ", ptr1 ptr2, CR,
       LF;                                                             0818
       IF f2 THEN *cadstr* _ *cadstr*, "    Host: ", ptr3 ptr4, CR,
       LF;                                                             0819
       END;                                                            0820
  getiadd(entrystr, 0, $ptr1, $ptr2);                                  0821
  IF cadstr.L > 0 OR ptr2[1] > ptr1[1] THEN                            0822
    BEGIN                                                              0823
    *statstr* _ *statstr*, "Mail Addresses: ", CR, LF;                 0824
    IF ptr2[1] > ptr1[1] THEN                                          0825
       BEGIN                                                           0970
       *statstr* _ *statstr*, "  Hardcopy Address: ";                  0826
       cetcapp($ptr1, $ptr2, &statstr);                                0972
       *statstr* _ *statstr*, CR, LF;                                  0973
       END;                                                            0971
    *statstr* _ *statstr*, *cadstr*;                                   0827
    END;                                                               0828
  getiverify(entrystr, 0, $ptr1, $ptr2);                               0829
  *statstr* _ *statstr*, ptr1 ptr2, CR, LF;                            0830
  IF getiphone(entrystr, 0, $ptr1, $ptr2) THEN                         0831
    BEGIN                                                              0832
    *statstr* _ *statstr*, "Phone: ", ptr1 ptr2, CR, LF;               0833
    END;                                                               0834
  IF getifunction(entrystr, 0, $ptr1, $ptr2) THEN                      0835
    BEGIN                                                              0836
    *statstr* _ *statstr*, "Function: ", ptr1 ptr2, CR, LF;            0837
    END;                                                               0838
  IF identwheel AND geticapability(entrystr, 0, $ptr1, $ptr2) THEN
                                                                       0839
                                                                       0840
    BEGIN
    *statstr* _ *statstr*, "Capabilities: ", ptr1 ptr2, CR, LF;
                                                                       0841
                                                                       0842
    END;                                                               0843
  IF getisubcol(entrystr, 0, $ptr1, $ptr2) THEN                        0844
    BEGIN
    *statstr* _ *statstr*, "Sub-Collection: ", ptr1 ptr2, CR, LF;
                                                                       0845
                                                                       0846
    END;                                                               0847
  IF getidelivery(entrystr, 0, $ptr1, $ptr2) THEN                      0848
    BEGIN                                                              0849
    *statstr* _ *statstr*, "Delivery: ", ptr1 ptr2, CR, LF;            0850
    END;                                                               0851
  IF getimcmnts(entrystr, 0, $ptr1, $ptr2) THEN                        0852
    BEGIN                                                              0853
    *statstr* _ *statstr*, "Comments: ", ptr1 ptr2, CR, LF;            0854
    END;                                                               0855
  RETURN;
  END.
                                                                       0856
                                                                       0857
                                                                       0254
```

FINISH of psjournal

PSSYNGEN

```
< NLS, PSSYNGEN.NLS;3, >, 19-NOV-74 22:52 DSM ;;;;
FILE pssyngen % L10 <rel-nls>pssyngen %% (110,) (rel-nls,pssyngen.rel,)%                02
                                                                                        03
% x - level routines for syntax generating subsystem %                                019
(xkshow) PROCEDURE                                                                      020
   % FORMALS %                                                                         021
      (resultptr, parsemode, type, ent, qual);                                          022
   LOCAL adr;                                                                           023
   REF resultptr, type, ent, qual;                                                     0171
   csstkx _ csstkb _ -1;                                                                024
   CASE parsemode OF                                                                    025
      = parsing:                                                                        026
        BEGIN                                                                           027
        CASE qual OF                                                                    028
           = 95 %- all -%:  cmdctl _ 0;                                                 029
           = 172 %- dnls -%:  cmdctl _ 1;                                               030
           = 173 %- tnls -%:  cmdctl _ 2;                                               031
           ENDCASE cmdctl _ 0;                                                          032
        CASE type OF                                                                    033
           = 174 %- command -%:                                                         034
              BEGIN                                                                     035
              fbctl( typenulllit );                                                     036
              ckshwcmd( ent );                                                          037
              fbctl( addcalit );                                                        038
              END;                                                                      039
           = 175 %- rule -%:                                                            040
              BEGIN                                                                     041
              IF NOT (adr _ gtadrs( &ent ) ) THEN                                       042
                 err($"Invalid command name");                                          043
              fbctl( typenulllit );                                                     044
              ckshwrul( adr );                                                          045
              fbctl( addcalit );                                                        046
              END;                                                                      047
           = 176 %- subsystem -%:                                                       048
              BEGIN                                                                     049
              fbctl( typenulllit );                                                     050
              ckshwsub( ent );                                                          051
              fbctl( addcalit );                                                        052
              END;                                                                      053
           ENDCASE;                                                                     054
        END;                                                                            055
      ENDCASE;                                                                          056
   RETURN( &resultptr );
   END.                                                                                 057
                                                                                        058
   %%
```

```
(xkcopy) PROCEDURE                                                    059
   % FORMALS %                                                        060
      (resultptr, parsemode, type, ent, qual, dest, lvl);             061
   LOCAL adr;                                                         062
   REF resultptr, type, ent, qual, dest, lvl;                         063
   csstkx _ csstkb _ -1;                                             0172
   CASE parsemode OF                                                  064
      = parsing:                                                      065
        BEGIN                                                         066
        CASE qual OF                                                  067
           = 95 %- all -%:  cmdctl _ 0;                               068
           = 172 %- dnls -%:  cmdctl _ 1;                             069
           = 173 %- tnls -%:  cmdctl _ 2;                             070
           ENDCASE cmdctl _ 0;                                        071
        CASE type OF                                                  072
           = 174 %- command -%:                                       073
              BEGIN                                                   074
              curmkr _ ckcopcmd( dest, lvl, ent );                    075
              curmkr[1] _ 1;                                          076
              END;                                                    077
           = 175 %- rule -%:                                          078
              BEGIN                                                   079
              IF NOT (adr _ gtadrs( &ent ) ) THEN                     080
                 err($"Invalid command name");                        081
              curmkr _ ckcoprul( dest, lvl, adr );                    082
              curmkr[1] _ 1;                                          083
              END;                                                    084
           = 176 %- subsystem -%:                                     085
              BEGIN                                                   086
              curmkr _ ckcopsub( dest, lvl, ent );                    087
              curmkr[1] _ 1;                                          088
              END;                                                    089
           ENDCASE;                                                   090
        dpset(dspstrc, curmkr, endfil, getnxt(curmkr));               091
        END;                                                          092
     ENDCASE;                                                         093
   RETURN( &resultptr );                                              094
   END.
                                                                      095
   %%                                                                096
```

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

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

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

PSSYSTEM

```
< NLS, PSSYSTEM.NLS;7, >, 9-MAR-77 15:55 JDH ;;;;
FILE pssystem % L10 <rel-nls>pssystem %% (L10,) (rel-nls,pssystem.rel,)
%                                                                    0977
% external label for ddt breakpoint %                               0978
   EXTERNAL brkhere;
% SUPERVISOR SUBSYSTEM %
   (xgoto) PROC( % GOTO execution routine %
      % FORMAL ARGUMENTS %
         resultptr,     % ptr to result record %
         parsemode,     % parsing mode %
         subsysptr,     % ptr to subsys name record %
         dispatchptr,   % ptr to subsystem dispatch pointer %
         exflagptr);    % ptr to EXECUTE flag record %
      % RETURNS %
         % 1) resultptr for TRUE return %
      % ABNORMAL RETURNS %
         % call to err if sbstack overflows %
      LOCAL % VARIABLES %
         entry,% ptr to new subsystem stack entry %
         count,% command repetition count %
         nptr,  % ptr to subsystem name %
         gptr;  % ptr to location in grammar %
      REF % VARIABLES %
         entry,
         dispatchptr,
         resultptr,
         subsysptr,
         exflagptr,
         nptr,
         gptr;
      %------------%
      CASE parsemode OF
         = parsing:
            BEGIN
            IF NOT exflagptr AND subsysptr = 117 %- tenex % THEN
            xgoexec()
            ELSE
               BEGIN
               count _ IF exflagptr THEN 1 ELSE -1;
               &nptr _ subsysptr;
               % set the grammar address according to which subsystem
               we're going to %
                  &gptr _ dispatchptr;
               % build a new subsystem stack entry if there is room %

                  IF sbstkx < $sbstksize
                     THEN
                        BEGIN
                        % save the current subsystem mode and set it to
                        rentry %
                           &entry _ $sbstack + sbstkx - $sbentsize;
                           entry.sbpmode _ entry.sbmode;
                           entry.sbmode _ sbrentry;
                        % set up a new substack entry %
                           &entry _ &entry + $sbentsize;
                           entry.sbmode _ sbstart;
```

```
                        entry.sbptr _ &gptr;                                    056
                        entry.sbnptr _ &nptr;                                   057
                        entry.sbcount _ count;                                  058
                        sbstkx _ sbstkx + $sbentsize;                           059
                     END                                                        060
                  ELSE err( $"subsystem stack overflow: Please quit
                  out of at least one subsystem");                              061
               END;                                                             062
            END;                                                                063
      ENDCASE;                                                                  064
      RETURN (&resultptr);                                                      065
      END.

                                                                                066
(xquit) PROC( % QUIT execution routine %                                        067
      % FORMAL ARGUMENTS %                                                      068
         resultptr,     % ptr to result record %                               069
         parsemode,     % parsing mode %                                       070
         subsysptr);    % ptr to subsys name record %                          071
      % RETURNS %                                                              072
         % 1) resultptr for TRUE return %                                       073
      % ABNORMAL RETURNS %                                                     074
         % call to err if subsystem is not in subsys stack %                    075
      LOCAL % VARIABLES %                                                       076
         level, % level in sbstack to cut back to %                            077
         index, % loop var, temp level counter %                               078
         entry,% ptr to new subsystem stack entry %                            079
         nptr,  % ptr to subsystem name %                                      080
         gptr;  % ptr to location in grammar %                                 081
      LOCAL STRING                                                              082
         errmess[50];   % error diagnostic string %                            083
      REF % VARIABLES %                                                         084
         entry,                                                                 085
         resultptr,                                                             086
         subsysptr,                                                             087
         exflagptr,                                                             088
         nptr,                                                                  089
         gptr;                                                                  090
      %------------%                                                           091
      CASE parsemode OF                                                         092
         = parsing:                                                             093
            BEGIN                                                               094
            % reset the cueflag so prompts will come out properly %             095
               cueflg _ FALSE;                                                  096
            dpset(dspno, endfil, endfil, endfil);                               097
            % set level to mark how far the subsystem stack is to be
            popped, level is set to the anticipated value for sbstkx
            after we've quit out of 1/more subsytems  %                         098
               CASE (&nptr _ subsysptr) OF                                      099
                  = 0:  % default, exit current subsystem only %                0100
                     level _ sbstkx - $sbentsize; % cut top entry only %
                                                                                0101
                  = 23 %- nls -%:       % quit NLS altogether %                0102
                     level _ 0;                                                 0103
               ENDCASE                                                          0104
                  BEGIN % search back through sbstack for match with
                  given name %                                                  0105
```

```
            FOR level _ sbstkx-$sbentsize DOWN $sbentsize UNTIL <
            0 DO                                                    0106
               BEGIN                                                0107
               &entry _ $sbstack + level;                           0108
               IF entry.sbnptr = &nptr THEN                         0109
                  BEGIN                                             0110
                  level _ level + $sbentsize;                       0111
                  EXIT CASE;                                        0112
                  END;                                              0113
               END;                                                 0114
            % no match was found, so put out an error %             0115
               *errmess* _ *nptr*, " is not in your subsystem
               stack";                                              0116
               err( $errmess );                                     0117
            END;                                                    0118
         % proceeed down the subsystem stack marking subsystems to be
         exited %                                                   0119
            FOR index _ sbstkx-$sbentsize DOWN $sbentsize UNTIL <
            level DO                                                0120
               BEGIN                                                0121
               &entry _ $sbstack + index;                           0122
               entry.sbmode _ sbfinish; % set to exit from subsystem
               %                                                    0123
               END;                                                 0124
         END;                                                       0125
   ENDCASE;                                                         0126
   RETURN (&resultptr);                                             0127
   END.                                                             0128

(xksyntax) PROCEDURE % show syntax of a command %                   0913
   % FORMALS %                                                      0914
      (resultptr, parsemode, anode);                                0915
   LOCAL adr;                                                       0916
   REF resultptr, anode;                                            0917
   csstkx _ csstkb _ -1;                                            0930
   CASE parsemode OF                                                0918
      = parsing:                                                    0919
         BEGIN                                                      0920
         cmdctl _ IF nlmode = fulldisplay THEN 1 ELSE 2;            0921
         fbctl( typenulllit );                                      0922
         ckshwcmd( anode );                                         0923
         fbctl( addcalit );                                         0924
         END;                                                       0925
      ENDCASE;                                                      0926
   RETURN( &resultptr );                                            0927
   END.                                                             0928

   %%                                                               0929
```

```
(xsublist) PROC( % show current subsystem stack %                    0129
   % FORMAL ARGUMENTS %                                              0130
      resultptr,     % ptr to result record %                        0131
      parsemode);    % parsing mode %                                0132
   LOCAL % variables %                                               0133
      entryptr;      % ptr to substack entry %                       0134
   LOCAL STRING str[300];                                            0135
   REF                                                               0136
      entryptr,                                                      0137
      resultptr;                                                     0138
   %----------%                                                      0139
   CASE parsemode OF                                                 0140
      = parsing:                                                     0141
         BEGIN                                                       0142
         % initialize display string %                               0143
            *str* _ "*** Subsystem Stack: (current one first) ***
            ";                                                       0144
         % append subsystem names to str %                           0145
            FOR &entryptr _ $sbstack+sbstkx-$sbentsize DOWN
            $sbentsize UNTIL < $sbstack DO                           0146
               *str* _ *str*, *[entryptr.sbnptr]*, EOL;              0147
         % put a trailer on the string %                             0148
            *str* _ *str*, "***";                                    0149
         % output the string %                                       0150
            fbctl( typecalit, $str );                                0151
         END;                                                        0152
   ENDCASE;                                                          0153
   RETURN (&resultptr);                                              0154
   END.
                                                                     0155
(xsubcurrent) PROC( % show current subsystem %                       0156
   % FORMAL ARGUMENTS %                                              0157
      resultptr,     % ptr to result record %                        0158
      parsemode);    % parsing mode %                                0159
   LOCAL STRING str[40];                                             0162
   REF                                                               0163
      resultptr;                                                     0165
   %----------%                                                      0166
   CASE parsemode OF                                                 0167
      = parsing:                                                     0168
         BEGIN                                                       0169
         % fetch subsystem name to str %                             0170
            xgtcrsb( $str );                                         0539
         % output the string %                                       0173
            fbctl( typecalit, $str );                                0174
         END;                                                        0175
   ENDCASE;                                                          0176
   RETURN (&resultptr);                                              0177
   END.
                                                                     0178
(xgtcrsb) PROC( % get current subsystem name to string %             0516
   % FORMAL ARGUMENTS %                                              0517
      astr); % adr string to get current name %                      0518
   LOCAL % variables %                                               0520
      entryptr;      % ptr to substack entry %                       0521
   REF                                                               0523
```

```
      entryptr,                                                    0524
      astr;                                                        0525
   %----------%                                                    0526
   % fetch subsystem name to str %                                 0530
      &entryptr _ $sbstack+sbstkx-$sbentsize;                      0531
      *astr* _ *[entryptr.sbnptr]*;                                0532
   RETURN;                                                         0537
   END.
                                                                   0538
(abortsubsystem) PROC( % aborts execution of a subsystem, prior to
normal termination.  Accomplishes what would be accomplished as if
the user had typed in QUIT CA %                                    0179
   % FORMAL ARGUMENTS %                                            0180
      errstrptr);    % ptr to diagnostic string %                 0181
   LOCAL % variables %                                             0182
      entry; % ptr to subsystem stack entry %                      0183
   REF errstrptr, entry;                                           0184
   % --------------- %                                            0185
   % change the mode of the current subsystem stack entry to exit the
   subsystem %                                                     0186
      &entry _ $sbstack + sbstkx - $sbentsize;                     0187
      entry.sbmode _ sbfinish;                                     0188
   % display the diagnostic string, generatiing a SIGNAL which causes
   us to exit the current subsystem after involing the termination
   rule (if any ) %                                                0189
      err( &errstrptr );                                           0190
   END.
                                                                   0191
%Jump%                                                             0711
   (xjump) %Execute Jump Command%                                  0712
      PROCEDURE                                                    0713
         %FORMALS%                                                 0714
            (result,          %result record%                      0715
            parsemode,        %parsing, backup, cleanup%           0716
            entity,           %type of load%                       0717
            destination,      %dest record%                        0718
            vs);              %view specs%                         0719
            REF                                                    0720
               result, entity, destination, vs;                    0721
         LOCAL da, start, oldda, fileno, linkp, vsupdate, adstr[40];
                                                                   0722
         REF da, oldda, linkp;                                     0723
         LOCAL TEXT POINTER t1, t2, csp;                           0724
         LOCAL STRING locstr[200];                                 0725
      %----------------------%                                     0726
      CASE parsemode OF                                            0727
         = parsing:                                                0728
            BEGIN                                                  0729
            % set so ^O won't clear output buffer %               01064
               rubnocob _ TRUE;                                  01065
            &da _ cspupdate _ lda();                               0730
            csp _ da.dacsp;  csp[1] _ da.daccnt;                   0731
            IF entity # 25 %- itemnovs -% THEN                     0994
               BEGIN                                               0995
               cspvs _ da.davspec;                                 0732
               cspvs[1] _ da.davspc2;                              0733
```

```
            cspcacode _ da.dacacode;                                    0734
            cspusqcod _ da.dausqcod;                                    0735
            END;                                                        0996
         vsupdate _ TRUE;                                               0736
         start _ &destination;                                          0737
         CASE entity OF                                                 0738
            = 24 %- item -%:                                            0739
               BEGIN                                                    0740
               (xj0):                                                   0741
               IF cspupdate THEN                                        0742
                  BEGIN                                                 0743
                  curmkr _ destination;                                 0744
                  curmkr[1] _ IF nlmode = fulldisplay THEN 1 ELSE
                  destination[1];                                       0745
                  END;                                                  0746
               IF vsupdate THEN                                         0747
                  BEGIN                                                 0748
                  cspvs     _ vs.vs1;                                   0749
                  cspvs[1]  _ vs.vs2;                                   0750
                  cspcacode _ vs.vscacode;                              0751
                  cspusqcod _ vs.vsusqcod;                              0752
                  END;                                                  0753
               dpset(dspjpf, curmkr, endfil, endfil);                   0754
               RETURN(&result);                                         0755
               END;                                                     0756
            = 25 %- itemnovs -%:                                        0931
               BEGIN                                                    0932
               vsupdate _ FALSE;                                        0933
               GOTO xj0;                                                0934
               END;                                                     0935
            = 26 %- successor -%:          *locstr* _ ".s";             0757
            = 27 %- predecessor -%:        *locstr* _ ".p";             0758
            = 28 %- up -%:                 *locstr* _ ".u";             0759
            = 29 %- down -%:               *locstr* _ ".d";             0760
            = 30 %- head -%:               *locstr* _ ".h";             0761
            = 31 %- tail -%:               *locstr* _ ".t";             0762
            = 32 %- end -%:                *locstr* _ ".e";             0763
            = 33 %- back -%:               *locstr* _ ".b";             0764
            = 34 %- next -%:               *locstr* _ ".n";             0765
            = 35 %- origin -%:             *locstr* _ ".o";             0766
            = 36 %- filereturn -%:                                      0767
               BEGIN                                                    0768
               curmkr _ destination.retstid;                            0769
               FIND SF(*[destination.retfn]*) ^t1;                      0770
               CASE lnbfls( $t1, 0, $locstr) OF                         0771
                  = lhostn: NULL;                                       0772
                  ENDCASE                                               0773
                     err($"Remote File Manipulations Not
                     Implemented Yet");                                 0774
               curmkr.stfile _ cloafil($locstr);                        0775
               curmkr[1] _ destination.retcc;                           0776
               vs _ destination.retvs1;                                 0777
               vs[1] _ destination.retvs2;                              0778
               vs.vscacode _ cspcacode;                                 0779
               vs.vsusqcod _ cspusqcod;                                 0780
               usesrr _ destination.retsrr;                             0781
```

```
                destination _ curmkr;                                   0782
                destination[1] _ curmkr[1];                             0783
                GOTO xj0;                                               0784
                END;                                                    0785
             = 37 %- return -%:                                         0786
                BEGIN                                                   0787
                curmkr _ destination.retstid;                           0788
                curmkr[1] _ destination.retcc;                          0789
                vs _ destination.retvs1;                                0790
                vs[1] _ destination.retvs2;                             0791
                vs.vscacode _ cspcacode;                                0792
                vs.vsusqcod _ cspusqcod;                                0793
                destination _ curmkr;                                   0794
                destination[1] _ curmkr[1];                             0795
                GOTO xj0;                                               0796
                END;                                                    0797
             = 8 %- link -%:                                            0798
                BEGIN                                                   0799
                vsupdate _ FALSE;                                       0800
                IF destination.stastr THEN                              0801
                   BEGIN                                                0802
                   start _ $csp;                                        0803
                   t1 _ destination; t1[1] _ destination[1];            0804
                   t2 _ [&destination+d2sel];                           0805
                      t2[1] _ [&destination+d2sel+1];                   0806
                   END                                                  0807
                ELSE                                                    0808
                   BEGIN                                                0809
                   lnkprs( &destination, $adstr);                       0810
                   t1 _ adstr[ls]; t1[1] _ adstr[ls+1];                 0811
                   t2 _ adstr[le]; t2[1] _ adstr[le+1];                 0812
                   END;                                                 0813
                GOTO xj2;                                               0814
                END;                                                    0815
             = 18 %- name -%:                                           0816
                BEGIN                                                   0817
                t1 _ destination;                                       0818
                t1[1] _ [&destination+d2sel+1];                         0819
                *locstr* _ destination t1;                              0820
                start _ $csp;                                           0821
                END;                                                    0822
             = 38 %- filename -%:                                       0906
                BEGIN                                                   0907
                t1 _ destination;                                       0908
                t1[1] _ [&destination+d2sel+1];                         0909
                *locstr* _ '(, destination t1, ",)";                    0910
                start _ $csp;                                           0911
                END;                                                    0912
             = 15 %- file -%:                                           0823
                BEGIN                                                   0824
                lnkprs(&destination, $adstr);                           0825
                t1 _ adstr[ls];  t1[1] _ adstr[ls+1];                   0826
                t2 _ adstr[fe];  t2[1] _ adstr[fe+1];                   0827
                *locstr* _ t1 t2, ",)";                                 0828
                start _ $csp;                                           0829
                END;                                                    0830
```

```
        = 39 %- firstname -%:                                          0831
           BEGIN                                                       0832
           t1 _ destination;                                           0833
           t1[1] _ [&destination+d2sel+1];                             0834
           *locstr* _ ".o*", destination t1;                           0835
           start _ $csp;                                               0836
           END;                                                        0837
        = 40 %- nextname -%:                                           0838
           BEGIN                                                       0839
           t1 _ destination;                                           0840
           t1[1] _ [&destination+d2sel+1];                             0841
           *locstr* _ '*, destination t1;                              0842
           start _ $csp;                                               0843
           END;                                                        0844
        = 41 %- extname -%:                                            0987
           BEGIN                                                       0988
           t1 _ destination;                                           0989
           t1[1] _ [&destination+d2sel+1];                             0990
           *locstr* _ '$, destination t1;                              0991
           start _ $csp;                                               0992
           END;                                                        0993
        = 42 %- firstcontent -%:                                       0852
           BEGIN                                                       0853
           t1 _ destination;                                           0854
           t1[1] _ [&destination+d2sel+1];                             0855
           *locstr* _ ".o", '", destination t1, '", "=C";              0856
           start _ $csp;                                               0857
           END;                                                        0858
        = 43 %- nextcontent -%:                                        0859
           BEGIN                                                       0860
           t1 _ destination;                                           0861
           t1[1] _ [&destination+d2sel+1];                             0862
           *locstr* _ ".n", '", destination t1, '", "=C";              0863
           start _ $csp;                                               0864
           END;                                                        0865
        = 44 %- firstword -%:                                          0866
           BEGIN                                                       0867
           t1 _ destination;                                           0868
           t1[1] _ [&destination+d2sel+1];                             0869
           *locstr* _ ".o", '", destination t1, '", "=W";              0870
           start _ $csp;                                               0871
           END;                                                        0872
        = 45 %- nextword -%:                                           0873
           BEGIN                                                       0874
           t1 _ destination;                                           0875
           t1[1] _ [&destination+d2sel+1];                             0876
           *locstr* _ ".n", '", destination t1, '", "=W";              0877
           start _ $csp;                                               0878
           END;                                                        0879
      ENDCASE err(notyet);                                             0880
    (xj1): %set up text pointers and evaluate address
    expression%                                                        0881
    FIND SF(*locstr*) ^t1 SE(*locstr*) ^t2;                            0882
    (xj2): %assuming the text pointers are set up, evaluate
    address expression%                                                0883
    IF vsupdate THEN                                                   0884
```

```
                BEGIN                                                   0885
                da.davspec _ vs;                                        0886
                da.davspc2 _ vs[1];                                     0887
                da.dacacode _ vs.vscacode;                              0888
                da.dausqcod _ vs.vsusqcod;                              0889
                END;                                                    0890
             vs _ caddexp($t1, $t2, &da, start : vs[1], vs.vscacode,
             vs.vsusqcod, usesrr);                                      0891
             IF vsupdate THEN                                           0892
                BEGIN                                                   0893
                da.davspec _ cspvs;                                     0894
                da.davspc2 _ cspvs[1];                                  0895
                da.dacacode _ cspcacode;                                0896
                da.dausqcod _ cspusqcod;                                0897
                END;                                                    0898
             vsupdate _ TRUE;                                           0899
             destination _ destination[d2sel] _ [start];                0900
                destination[1] _ destination[d2sel+1] _ [start+1];
                                                                        0901
             GOTO xj0;                                                  0902
             END;                                                       0903
          ENDCASE RETURN(&result);                                      0904
       END.
                                                                        0905
    (updcsp) PROCEDURE (result, pmode);                                01066
       REF result;                                                      01067
       CASE pmode OF                                                    01068
          = parsing:  cspupdate _ lda();                                01069
          ENDCASE;                                                      01070
       RETURN(&result);                                                 01071
       END.                                                             01072
% TENEX SUBSYSTEM %                                                    0192
   (xgoexec) PROCEDURE;          %goto exec command%                    0193
      <AUXCOD, gofork>(IF tops20flag THEN $"<SYSTEM>EXEC.EXE"  ELSE
      $"<SYSTEM>EXEC.SAV", -1, -1, 4B10);                               0194
      dpset(dspallf, endfil, endfil, endfil);                           0195
      RETURN END.
                                                                        0196
% UTILITY SUBSYSTEM %                                                  0197
   % this file contains the "x" routines for support of the LIBENT
   parser%                                                              0198
   % not called                                                         0936
   (xutilinit) PROCEDURE( %%initialize stuff for utility functions%%
                                                                        0937
      %% FORMAL ARGUMENTS %%                                           0938
      resultptr,          %%ptr to result record%%                      0939
      pmode);             %%parsing mode%%                              0940
      REF resultptr;                                                    0941
      LOCAL STRING uname[10];                                           0942
      %%--------------%%                                               0943
      CASE pmode OF                                                     0944
         = parsing:                                                     0945
            BEGIN                                                       0946
            *uname* _ "UTILTY";                                         0947
            nlssbn _ getsbn($uname); %%convert to sixbit%%              0948
            !setnm(nlssbn);                                             0949
```

```
        libflg _ oldflg _ liblod _ jdfi _ jdid _ jdno _ 0;
        %%library functions, parameter strings null%%                    0950
        END;                                                             0951
   ENDCASE;                                                              0952
   RETURN( &resultptr);                                                  0953
   END.
                                                                         0954
%                                                                        0955
(xjoutil) PROCEDURE( %run the journal - maintenance mode%               01006
   % FORMAL ARGUMENTS %                                                 01007
   resultptr,        %ptr to result record%                             01008
   pmode);   %parsing mode%                                             01009
   REF resultptr;                                                       01010
   %--------------%                                                     01011
   CASE pmode OF                                                        01012
      = parsing:                                                        01013
        BEGIN                                                           01014
        jnlrun(); % roll in JNLDEL file and start processing. %
                                                                        01015
        END;                                                            01016
      = cleanup:                                                        01017
        BEGIN                                                           01018
        IF jdid THEN freestring(jdid := 0, $dspblk);                    01019
        IF jdfi THEN freestring(jdfi := 0, $dspblk);                    01020
        IF jdno THEN freestring(jdno := 0, $dspblk);                    01021
        libflg _ 0;                                                     01022
        END;                                                            01023
      ENDCASE;                                                          01024
   RETURN( &resultptr);                                                 01025
   END.
                                                                        01026
(jnlout)PROCEDURE; % Roll out JNLDEL file %                              0998
   IF jnlprog THEN % Reset program buffer. %                             0999
      BEGIN                                                            01000
      gpgmrst(); % Reset stack %                                       01001
      jnlprog_FALSE;                                                   01002
      END;                                                             01003
   RETURN;                                                             01004
   END.
                                                                       01005
(xcutil) PROCEDURE(  %run tasks%                                         0232
   % FORMAL ARGUMENTS %                                                 0233
   resultptr,  %ptr to result record%                                   0234
   pmode,    %parsing mode%                                             0235
   umode);    %flags to be passed to nlsutility%                        0236
   REF resultptr, umode;                                                0237
   %simply call utility function with prior specified mode%             0238
      CASE pmode OF                                                     0239
         = parsing:                                                     0240
           BEGIN                                                        0241
           typeas($"
             *** Running Tasks ***");                                   0242
           nlsutility(CASE umode OF                                     0243
             = 46 %- detached -%: 4;                                    0310
             = 15 %- file -%: 5;                                        0309
             = 47 %- tty -%: 6;                                         0311
```

```
                ENDCASE 5);                                                    0312
              END;                                                             0244
        ENDCASE;                                                               0245
        RETURN( &resultptr);                                                   0246
        END.                                                                   0247
(xsetjou) PROCEDURE(  %set flags for journal run%                              0248
    % FORMAL ARGUMENTS %                                                       0249
    resultptr,          %ptr to result record%                                 0250
    pmode,              %parsing mode%                                         0251
    option);            %pointer to option specified%                          0252
    REF resultptr, option;                                                     0253
    CASE pmode OF                                                              0254
       = parsing:                                                              0255
          BEGIN                                                                0256
          CASE option OF                                                       0257
             = 48 %- auto -%: libflg _libflg .V 20B;                           0258
             = 49 %- continue -%:     libflg _libflg .V 10B;                  0259
             = 50 %- on -%:           libflg _libflg .V 2B;                   0260
             = 51 %- recover -%:      libflg _libflg .V 1B;                   0262
             = 52 %- slinker -%:      libflg _libflg .V 4B;                   0263
             = 53 %- update -%:       libflg _libflg .V 40B;                  0264
             ENDCASE  typeas($"invalid option specified");                     0265
          END;                                                                 0266
     ENDCASE;                                                                  0267
     RETURN( &resultptr);                                                      0268
     END.                                                                      0269

(xjnlload) PROCEDURE(  %load JNLDEL into programs buffer%                      01027
    % FORMAL ARGUMENTS %                                                       01028
    resultptr,          %ptr to result record%                                 01029
    pmode);             %parsing mode%                                         01030
    REF resultptr, option;                                                     01031
    CASE pmode OF                                                              01032
       = parsing:                                                              01033
          BEGIN                                                                01034
          jnlin();                                                             01035
          END;                                                                 01036
     ENDCASE;                                                                  01037
     RETURN( &resultptr);                                                      01038
     END.                                                                      01039

(xjnldelete) PROCEDURE(  %delete JNLDEL from programs buffer%                  01040
    % FORMAL ARGUMENTS %                                                       01041
    resultptr,          %ptr to result record%                                 01042
    pmode);             %parsing mode%                                         01043
    REF resultptr, option;                                                     01044
    CASE pmode OF                                                              01045
       = parsing:                                                              01046
          BEGIN                                                                01047
          jnlout();                                                            01048
          END;                                                                 01049
     ENDCASE;                                                                  01050
     RETURN( &resultptr);                                                      01051
     END.                                                                      01052

(xsetpar) PROCEDURE(  %set flags for partial journal delivery%                 0467
```

```
    % FORMAL ARGUMENTS %                                                    0468
    resultptr,          %ptr to result record%                              0469
    pmode,              %parsing mode%                                      0470
    entity,             %pointer to entity selected%                        0471
    option);            %pointer to option specified%                       0472
    LOCAL TEXT POINTER z1, z2;                                              0515
    REF resultptr, entity, option;                                          0473
    CASE pmode OF                                                           0474
       = parsing:                                                           0475
         BEGIN                                                              0476
         IF option # 54 %+ clear +% THEN                                    0477
            BEGIN                                                           0478
            z1 _ entity;                                                    0479
            z1[1] _ [&entity + 1];                                          0480
            z2 _  [&entity  + 2];                                           0481
            z2[1] _ [&entity + 3];                                          0482
            END;                                                            0483
         CASE option OF                                                     0484
            = 54 %- clear -%:                                               0485
               BEGIN                                                        0486
               oldflg _ 0;                                                  0487
               IF jdid THEN freestring(jdid := 0, $dspblk);                 0488
               IF jdfi THEN freestring(jdfi := 0, $dspblk);                 0489
               IF jdno THEN freestring(jdno := 0, $dspblk);                 0490
               END;                                                         0491
            = 55 %- idents -%:                                              0492
               BEGIN                                                        0493
               jdid _ getstring(1000, $dspblk);                             0494
               *[jdid]* _ z1 z2;                                            0495
               oldflg _ oldflg .V 200B;                                     0496
               END;                                                         0497
            = 56 %- files -%:                                               0498
               BEGIN                                                        0499
               jdfi _ getstring(1000, $dspblk);                             0500
               *[jdfi]* _ z1 z2;                                            0501
               oldflg _ oldflg .V 100B;                                     0502
               END;                                                         0503
            = 11 %- number -%:                                              0504
               BEGIN                                                        0505
               jdno _ getstring(1000, $dspblk);                             0506
               *[jdno]* _ z1 z2;                                            0507
               oldflg _oldflg .V 400B;                                      0508
               END;                                                         0509
            ENDCASE  dismes(2,$"invalid option specified");                 0510
         END;                                                               0511
    ENDCASE;                                                                0512
    RETURN ( &resultptr);                                                   0513
    END.
                                                                            0514
(xsetld) PROCEDURE(  %set load average cutoff value%                        0379
    % FORMAL ARGUMENTS %                                                    0380
    resultptr,          %ptr to result record%                              0381
    pmode,              %parsing mode%                                      0382
    param);             %pointer to selection record%                       0383
    LOCAL mlav[2], tp;                                                      0384
    LOCAL STRING lavstr[30];                                                0385
```

```
   REF resultptr, param, tp;                                          0386
   CASE pmode OF                                                      0387
      = parsing:                                                      0388
         BEGIN                                                        0389
         &tp _ &param + d2sel;                                        0390
         *lavstr* _ param tp;                                         0391
         nfloat($lavstr, $mlav, $mlav + 1);                           0392
         oljmlav _ mlav;                                              0393
         liblod _ TRUE; % set flag which syas load average has been
         set by hand; JOUTIL checks it to see if it has to set the
         load average cutoff. %                                       0394
         END;                                                         0395
      ENDCASE;                                                        0396
   RETURN ( &resultptr);                                              0397
   END.
                                                                      0398
(xpriority)  % set priority only if person can enable %               0413
   PROCEDURE (resultptr, parsemode, priorval);                        0414
   LOCAL value, char;                                                 0415
   LOCAL TEXT POINTER z1, z2;                                         0416
   LOCAL STRING pristr[10];                                           0417
   REF resultptr, priorval;                                           0418
   CASE parsemode OF                                                  0419
      = parsing:                                                      0420
         BEGIN                                                        0421
         % get information out of the parameter records %             0422
            z1 _ priorval;                                            0423
            z1[1] _ [&priorval + 1];                                  0424
            z2 _ [&priorval + 2];                                     0425
            z2[1] _ [&priorval + 3];                                  0426
            *pristr* _ z1 z2;                                         0427
            % Remove blanks %                                        0428
            IF NOT FIND SF(*pristr*) $NP ^z1 $D ^z2 THEN              0429
               err($"Illegal priority value");                        0430
            *pristr* _ z1 z2;                                         0431
         % check the validity of the value %                         0432
            CASE pristr.L OF                                          0433
               = 0: value _ 202B;                                     0434
               = 1:                                                   0435
                  IF *pristr*[1] = '0 THEN value _ 0                  0436
                  ELSE REPEAT CASE(4); % Force error %                0437
               = 3:                                                   0438
                  BEGIN                                               0439
                  IF (char _ *pristr*[1]) IN ['1, '3] THEN            0440
                     value _ (char - '0) * 100B                       0441
                  ELSE REPEAT CASE(4); % Force error %                0442
                  IF *pristr*[2] # '0 THEN REPEATCASE(4);             0443
                  IF (char _ *pristr*[3]) IN ['1, '3] THEN            0444
                     value _ (char - '0) + value                      0445
                  ELSE REPEAT CASE(4); % Force error %                0446
                  END;                                                0447
               ENDCASE err($"Illegal priority value");                0448
         % set the priority %                                         0449
            setpriority ( value );                                    0450
         END;                                                         0451
   ENDCASE;                                                           0452
```

```
   RETURN( &resultptr );                                                0453
   END.
                                                                        0454
(xxddt)      % go to DDT %                                              0540
   PROCEDURE (resultptr, parsemode);                                    0541
   REF resultptr;                                                       0545
   CASE parsemode OF                                                    0546
      = parsing:  ddt();                                                0547
      ENDCASE;                                                          0579
   RETURN( &resultptr );                                                0580
   END.
                                                                        0581
(ddt)    % dummy procedure to get us into ddt %                         0966
   PROCEDURE;                                                           0967
   IF (tenex >= 13200) AND (nldevice = devlproc) THEN                   0968
      BEGIN                                                             0969
      !bout( dspjfn, lpesc);                                            0970
      !bout( dspjfn, lpnocoor);                                         0971
      END;                                                              0972
   (brkhere):                                                           0973
   IF (tenex >= 13200) AND (nldevice = devlproc) THEN lpcmode();        0974
   RETURN;                                                              0975
   END.
                                                                        0976
(xxcheck) PROCEDURE  % Check results of running tasks %                 0586
   (resultptr, parsemode);                                              0593
   REF resultptr;                                                       0587
   CASE parsemode OF                                                    0588
      = parsing:  inptrf _ tskerrcnt;                                   0589
      ENDCASE;                                                          0590
   RETURN( &resultptr );                                                0591
   END.
                                                                        0592
(xxdetach) PROCEDURE (resultptr, parsemode, infile, outfile);           0594
   LOCAL                                                                0595
      rhosti, rhosto, inparam, outparam;                                0596
   LOCAL STRING                                                         0597
      instring[200], outstring[200];                                    0598
   REF resultptr, infile, outfile;                                      0599
   CASE parsemode OF                                                    0600
      = parsing:                                                        0601
         BEGIN                                                          0602
         rhosti _ rhosto _ lhostn;                                      0603
         inparam _ outparam _ FALSE;                                    0604
         IF infile THEN                                                 0605
            BEGIN                                                       0606
            rhosti_lnbfls(&infile, 0, $instring);                       0607
            inparam _ $instring;                                        0608
            END;                                                        0609
         IF outfile THEN                                                0610
            BEGIN                                                       0611
            rhosto_lnbfls(&outfile, 0, $outstring);                     0612
            outparam _ $outstring;                                      0613
            END;                                                        0614
         cxdetach( rhosti, inparam,  rhosto, outparam);                 0615
         END;                                                           0619
```

```
      ENDCASE;                                                          0616
    RETURN( &resultptr );                                               0617
    END.
                                                                        0618
% not called                                                           0956
(xsubnotimp)            %% subsystem not yet implemented %%           0957
  PROCEDURE (resultptr, parsemode);                                     0958
  REF resultptr;                                                        0959
  CASE parsemode OF                                                     0960
    = parsing: err($"subsystem not implemented yet.  Use QUIT command
    to return");                                                        0961
  ENDCASE;                                                              0962
  RETURN( &resultptr );                                                 0963
  END.
                                                                        0964
%                                                                      01053
                                                                       01054
                                                                       01055
%DUMMY X-ROUTINE%                                                      01056
  % Called to by-pass a CLI bug -- does nothing.  Used by SENDMAIL
  Interrogate command %                                                01057
                                                                       01058
  (xdummy) PROCEDURE (result, parsemode);                              01059
    REF result;                                                        01060
    RETURN (&result);                                                  01061
    END.                                                               01062
                                                                       01063
FINISH of pssystem                                                      0308
```

P SUPPORT

```
< NLS, PSUPPORT.NLS.16, >, 21-Dec-77 19:47 JDH ;;;;  % PARSER SUPPORT
CODE %
FILE psupport % L10 <rel-nls>psupport %% (L10,) (rel-nls,psupport.rel,)
%                                                                      02
% Declarations %                                                       03
   REGISTER r1=1, r2=2, r3=3, r4=4;                                    04
   REF msgda, rawchr, inpt, tda;                                       05
   DECLARE EXTERNAL cutpathstk = 999879;  %value for SIGNAL code%   0811
   DECLARE                                                             06
      nofile = 0, noct = 0,ctcsp = 1, ctfrz = 2,                       07
      ctcpfz = 3, ctmkr = 8, ctcmk = 9, ctcfm = 11, ctlcfm = 15,
      copyflag = 1, moveflag = 2;                                      08
                                                                       09
   DECLARE notyet = 7;                                                010
% PARSING FUNCTIONS %                                                 014
   % READS A SPACE %                                                  015
      (sp) PROC(curptr, parsemode, string);                            016
         % sp looks at the next inpt character, if it is a space, then
         the space is read and a true return is taken.  If the next
         character is not a space, then it is not read, and FALSE is
         returned %                                                    017
         REF curptr, string;                                           018
         %------------%                                               019
         CASE parsemode OF                                             020
            = parsing:                                                 021
               CASE lookc() OF                                         022
                  = SP:                                                023
                     inpt();                                           024
               ENDCASE RETURN (FALSE);                                 025
            = parsehelp:                                               026
               IF curptr.begnodeptr = curptr.curnodeptr THEN       01095
                  *string* _ "SP:";                                    027
            = parseqmark:                                             0677
               BEGIN                                                  0679
               *string* _ "<SPACE>";                                  0678
               RETURN;                                                0680
               END;                                                   0681
         ENDCASE;                                                      028
         RETURN (&curptr);                                             029
         END.
                                                                       030
                                                                       031
   % READS AN OPTION CHARACTER %                                       032
      (readoption) PROCEDURE( % parsing function which looks at the next
      input char.  If it is an option character, then it reads the char
      and returns TRUE, otherwise it returns FALSE %                   033
         % FORMAL ARGUMENTS %                                          034
            resultptr,  % ptr to the result record %                   035
            parsemode,  % parsing mode %                               036
            string);    % ptr to help string %                        037
         REF resultptr, string, inpt;                                  038
         % ------------- %                                            039
            CASE parsemode OF                                          040
               = parsing:                                              041
                  CASE lookc() OF                                      042
                     = optchar:                                        043
```

```
                    inpt();                                          044
                ENDCASE RETURN (FALSE);                              045
            = parsehelp:                                             046
               IF inprompts NOT= partprompts AND resultptr.begnodeptr
               = resultptr.curnodeptr THEN                           0981
                  *string* _ "OPT:";                                 0982
            = parseqmark:                                            0682
               BEGIN                                                 0683
               *string* _ "OPTION";                                  0684
               RETURN;                                               0685
               END;                                                  0686
            ENDCASE;                                                 048
         RETURN (&resultptr );                                       049
         END.
                                                                     050
% READS AN REPEAT CHARACTER %                                        051
   (readrepeat) PROCEDURE( % parsing function which looks at the next
   input char.  If it is an REPEAT character, then it reads the char
   and returns TRUE, otherwise it returns FALSE %                    052
      % FORMAL ARGUMENTS %                                           053
         resultptr,  % ptr to the result record %                    054
         parsemode,  % parsing mode %                                055
         string);    % ptr to help string %                          056
      REF resultptr, string;                                         057
      % -------------- %                                            058
         CASE parsemode OF                                           059
            = parsing:                                               060
               CASE lookc() OF                                       061
                  = rptchar:                                         062
                     inpt();                                         063
                  ENDCASE RETURN (FALSE);                            064
            = parsehelp:                                             065
               IF resultptr.begnodeptr = resultptr.curnodeptr THEN
                                                                     01086
                  *string* _ "RPT:";                                01088
            = parseqmark:                                            0687
               *string* _ "<REPEAT>";                                0689
            ENDCASE;                                                 067
         RETURN (&resultptr );                                       068
         END.
                                                                     069
% READS AN ANSWER CHARACTER %                                        070
   (answ) PROCEDURE( % reads next input char, returns TRUE if answer
   is yes, FALSE otherwise.  CA and Y denote YES, all other chars
   denote NO %                                                       071
      % FORMAL ARGUMENTS %                                           072
         resultptr,  % ptr to result record %                        073
         parsemode,  % interpreter parsing mode %                    074
         stringptr); % ptr to help string %                          075
      REF resultptr, stringptr, inpt;                                076
      % ---------------- %                                          077
      CASE parsemode OF                                              078
         = parsing:                                                  079
            BEGIN                                                    080
            cueflg _ FALSE;                                          081
            CASE lookc() OF                                          082
```

```
            = cachar, = optchar, = rptchar, = inschar:          083
               BEGIN                                            084
               inpt();                                          085
               needconfirm _ FALSE;                            01109
               RETURN (&resultptr);                             086
               END;                                             087
            = 'Y, ='y:                                          088
               BEGIN                                            089
               inpt();                                          090
               needconfirm _ FALSE;                            01110
               curchr _ CA;                                     0746
               RETURN (&resultptr);                             091
               END;                                             092
            = CD:                                               093
               SIGNAL(cmdelete);                                094
            =BC, =BW:                                           095
               SIGNAL (popstate);                               096
            ENDCASE                                             097
               BEGIN                                           01084
               needconfirm _ TRUE;                             01083
               inpt();                                          098
               END;                                            01085
         RETURN (FALSE);                                        099
         END;                                                  0100
      = parsehelp:                                             0101
         IF resultptr.begnodeptr = resultptr.curnodeptr THEN   01077
            *stringptr* _ "Y/N:"                               0102
         ELSE *stringptr* _ NULL;                             01079
      = parseqmark:                                            0692
         BEGIN                                                 0693
         *stringptr* _ "Y for yes", 0, "N for no";            01081
         RETURN;                                               0695
         END;                                                  0696
      ENDCASE;                                                 0103
   RETURN (&resultptr);                                        0104
   END.
                                                               0105
(answer) PROCEDURE( % reads next input char, returns pointer to
string 0/1.  CA and Y denote YES, all other chars denote NO %  0106
   % FORMAL ARGUMENTS %                                        0107
      resultptr,  % ptr to result record %                     0108
      parsemode,  % interpreter parsing mode %                 0109
      stringptr); % ptr to help string %                       0110
   REF resultptr, stringptr, inpt;                             0111
   % ---------------- %                                       0112
   CASE parsemode OF                                           0113
      = parsing:                                               0114
         BEGIN                                                 0115
         cueflg _ FALSE;                                       0116
         needconfirm _ TRUE;                                   0985
         CASE inpt() OF                                        0117
            = cachar, = optchar, = rptchar, = inschar:         0118
               BEGIN                                           0119
               resultptr _ TRUE;                               0120
               END;                                            0121
            = 'Y, ='y:                                         0747
```

```
                BEGIN                                                   0748
                resultptr _ TRUE;                                       0749
                curchr _ CA;                                            0751
                END;                                                    0750
             = CD:                                                      0122
                SIGNAL(cmdelete);                                       0123
             =BC, =BW:                                                  0124
                SIGNAL (popstate);                                      0125
             ENDCASE                                                    0126
                resultptr _ FALSE;                                      0127
          RETURN (&resultptr);                                          0128
          END;                                                          0129
       = parsehelp:                                                     0130
          IF resultptr.begnodeptr = resultptr.curnodeptr THEN           01097
             *stringptr* _ "Y/N:";                                      0131
       = parseqmark:                                                    0697
          BEGIN                                                         0698
          *stringptr* _ "Y for yes", 0, "N for no";                     0699
          RETURN;                                                       0700
          END;                                                          0701
       ENDCASE;                                                         0132
    RETURN (&resultptr);                                                0133
    END.                                                                0134

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

```
            IF resultptr.begnodeptr = resultptr.curnodeptr THEN 01098
               *stringptr* _ "S:";                              01019
         = parseqmark:                                          01020
            BEGIN                                               01021
            *stringptr* _  "S for status";                      01022
            RETURN;                                             01023
            END;                                                01024
         ENDCASE;                                               01025
      RETURN (&resultptr);                                      01026
      END.
                                                                01027
% LOOKS FOR ANSWER CHARACTER %                                   0135
   (lookansw) PROCEDURE( % looks at next input char, returns TRUE if
   answer is yes, FALSE otherwise.  CA and Y denote YES, all other
   chars denote NO %                                            0136
      % FORMAL ARGUMENTS %                                      0137
         resultptr,  % ptr to result record %                   0138
         parsemode,  % interpreter parsing mode %               0139
         stringptr); % ptr to help string %                     0140
      REF resultptr, stringptr, inpt;                           0141
      % ---------------- %                                      0142
      CASE parsemode OF                                         0143
         = parsing:                                             0144
            BEGIN                                               0145
            cueflg _ FALSE;                                     0146
            CASE lookc() OF                                     0147
               = cachar, = optchar, = rptchar, = inschar:       0148
                  BEGIN                                         0149
                  RETURN (&resultptr);                          0150
                  END;                                          0151
               = 'Y, ='y:                                       0152
                  BEGIN                                         0153
                  inpt();                                       0154
                  curchr _ CA;                                  0745
                  RETURN (&resultptr);                          0155
                  END;                                          0156
               = CD:                                            0157
                  SIGNAL(cmdelete);                             0158
               =BC, =BW:                                        0159
                  SIGNAL (popstate);                            0160
               ENDCASE                                          0161
                  inpt();                                       0162
            RETURN (FALSE);                                     0163
            END;                                                0164
         = parsehelp:                                           0165
            IF resultptr.begnodeptr = resultptr.curnodeptr THEN 01099
               *stringptr* _ "Y/N:";                            0166
         = parseqmark:                                          0702
            BEGIN                                               0703
            *stringptr* _ "Y for yes", 0, "N for no";           0704
            RETURN;                                             0705
            END;                                                0706
         ENDCASE;                                               0167
      RETURN (&resultptr);                                      0168
      END.
                                                                0169
```

```
   (sublookansw) PROCEDURE( % looks at next input char, returns TRUE
   if answer is yes, FALSE otherwise.  CA and Y denote YES, all other
   chars denote NO %                                                  01037
      % FORMAL ARGUMENTS %                                            01038
         resultptr,  % ptr to result record %                         01039
         parsemode,  % interpreter parsing mode %                     01040
         stringptr); % ptr to help string %                           01041
      REF resultptr, stringptr, inpt;                                 01042
      % ---------------- %                                            01043
      CASE parsemode OF                                               01044
         = parsing:                                                   01045
            BEGIN                                                     01046
            cueflg _ FALSE;                                           01047
            CASE lookc() OF                                           01048
               = cachar, = optchar, = rptchar, = inschar:             01049
                  BEGIN                                               01050
                  RETURN (&resultptr);                                01051
                  END;                                                01052
               = 'Y, ='y:                                             01053
                  BEGIN                                               01054
                  inpt();                                             01055
                  curchr _ CA;                                        01056
                  RETURN (&resultptr);                                01057
                  END;                                                01058
               = CD:                                                  01059
                  SIGNAL(cmdelete);                                   01060
               =BC, =BW:                                              01061
                  SIGNAL (popstate);                                  01062
               ENDCASE                                                01063
                  NULL;                                               01064
            RETURN (FALSE);                                           01065
            END;                                                      01066
         = parsehelp:                                                 01067
            IF resultptr.begnodeptr = resultptr.curnodeptr THEN       01100
            *stringptr* _ "Y/N:";                                     01068
         = parseqmark:                                                01069
            BEGIN                                                     01070
            *stringptr* _ "Y for yes", 0, "N for no";                 01071
            RETURN;                                                   01072
            END;                                                      01073
         ENDCASE;                                                     01074
      RETURN (&resultptr);                                            01075
      END.

                                                                      01076
% READS A NON-CA CHARACTER %                                            0170
   (notca) PROC(curptr, parsemode, string);                            0171
      % notca looks at the next inpt character, if it is not a CA,
      then the character is read and a true return is taken.  If the
      next character is a CA, then it is not read, and FALSE is
      returned %                                                      0172
      REF curptr, string;                                             0173
      %------------%                                                  0174
      CASE parsemode OF                                               0175
         = parsing:                                                   0176
            CASE lookc() OF                                           0177
               # cachar:                                              0178
```

```
                inpt();                                                 0179
            ENDCASE RETURN (FALSE);                                     0180
         = parsehelp:                                                   0181
            IF curptr.begnodeptr = curptr.curnodeptr THEN              01101
               *string* _ "#CA:";                                       0182
         = parseqmark:                                                  0707
            BEGIN                                                       0708
            *string* _ "NOT <CA>";                                      0709
            RETURN;                                                     0710
            END;                                                        0711
      ENDCASE;                                                          0183
      RETURN (&curptr);                                                 0184
      END.
                                                                        0185
                                                                        0186
% READS A CA CHARACTER %                                               0187
   (readconfirm) PROC(curptr, parsemode, string);                       0188
      % readconfirm looks at the next inpt character, if it is a
      CA/REPEAT/INSERT, then it is read and a true return is taken
      else FALSE is returned %                                          0189
      REF curptr, string;                                               0190
      %------------%                                                   0191
      CASE parsemode OF                                                 0192
         = parsing:                                                     0193
            CASE lookc() OF                                             0194
               = cachar, = rptchar, = inschar:                          0195
                  BEGIN                                                 0196
                  % stick ptr to castring into result record %          0197
                     curptr _ $castr;                                   0198
                  % read over the CA %                                  0199
                     inpt();                                            0200
                  END;                                                  0201
            ENDCASE RETURN (FALSE);                                     0202
         = parsehelp:                                                   0203
            IF curptr.begnodeptr = curptr.curnodeptr THEN              01102
               *string* _ "OK:";                                        0204
         = parseqmark:                                                  0712
            BEGIN                                                       0713
            *string* _ "OK";                                            0714
            RETURN;                                                     0715
            END;                                                        0716
      ENDCASE;                                                          0205
      RETURN (&curptr);                                                 0206
      END.
                                                                        0207
                                                                        0208
   (readbug) PROC(curptr, parsemode, string);                           0209
      % readbug looks at the next inpt character, if it is a CA, then
      it is read and a true return is taken else FALSE is returned %
                                                                        0210
      REF curptr, string;                                               0211
      %------------%                                                   0212
      CASE parsemode OF                                                 0213
         = parsing:                                                     0214
            CASE lookc() OF                                             0215
               = cachar, = rptchar, = inschar:                          0216
```

```
                    BEGIN                                                   0217
                    % stick ptr to castring into result record %            0218
                       curptr _ $castr;                                     0219
                    % read over the CA %                                    0220
                       inpt();                                              0221
                    END;                                                    0222
                ENDCASE RETURN (FALSE);                                     0223
            = parsehelp:                                                    0224
                IF curptr.begnodeptr = curptr.curnodeptr THEN              01103
                   IF nlmode = fulldisplay THEN *string* _ "B:"             0225
                   ELSE *string* _ "OK:";                                   0655
            = parseqmark:                                                   0717
                BEGIN                                                       0718
                IF nlmode = fulldisplay THEN *string* _ "BUG"               0719
                ELSE *string* _ "OK";                                       0722
                RETURN;                                                     0720
                END;                                                        0721
        ENDCASE;                                                            0226
        RETURN (&curptr);                                                   0227
        END.

                                                                            0228
                                                                            0229
% LOOK FOR A CA CHARACTER %                                                 0230
    (lookconfirm) PROC(curptr, parsemode, string);                          0231
        % lookconfirm looks at the next inpt character, if it is a
        CA/REPEAT/INSERT, then a true return is taken else FALSE is
        returned %                                                          0232
        REF curptr, string;                                                 0233
        %------------%                                                     0234
        CASE parsemode OF                                                   0235
            = parsing:                                                      0236
                CASE lookc() OF                                             0237
                    = cachar, = rptchar, = inschar:                         0238
                        NULL;                                               0239
                ENDCASE RETURN (FALSE);                                     0240
            = parsehelp:                                                    0241
                IF curptr.begnodeptr = curptr.curnodeptr THEN              01104
                   *string* _ "OK:";                                        0242
            = parseqmark:                                                   0723
                BEGIN                                                       0724
                *string* _ "OK";                                            0725
                RETURN;                                                     0726
                END;                                                        0727
        ENDCASE;                                                            0243
        RETURN (&curptr);                                                   0244
        END.

                                                                            0245
                                                                            0246
    (lookbug) PROC(curptr, parsemode, string);                              0247
        % lookbug looks at the next inpt character, if it is a CA, then
        a true return is taken else FALSE is returned %                     0248
        REF curptr, string;                                                 0249
        %------------%                                                     0250
        CASE parsemode OF                                                   0251
            = parsing:                                                      0252
                CASE lookc() OF                                             0253
```

```
            = cachar, = rptchar, = inschar:                          0254
               NULL;                                                 0255
            ENDCASE RETURN (FALSE);                                  0256
         = parsehelp:                                                0257
            IF curptr.begnodeptr = curptr.curnodeptr THEN           01105
               IF nlmode = fulldisplay THEN *string* _ "B:"          0258
               ELSE *string* _ NULL;                                 0654
         = parseqmark:                                               0728
            BEGIN                                                    0729
            IF nlmode = fulldisplay THEN *string* _ "BUG"            0733
            ELSE *string* _ NULL;                                    0734
            RETURN;                                                  0731
            END;                                                     0732
         ENDCASE;                                                    0259
      RETURN (&curptr);                                              0260
      END.
                                                                     0261
                                                                     0262
% LOOK FOR A NUMBER %                                               0276
   (looknum) PROC(curptr, parsemode, string);                        0277
      % looknum looks at the next inpt character, if it is a digit,
      then a true return is taken else FALSE is returned %          0278
      REF curptr, string;                                            0279
      %------------%                                                 0280
      CASE parsemode OF                                              0281
         = parsing:                                                  0282
            CASE lookc() OF                                          0283
               IN ['0, '9]:                                          0284
                  NULL;                                              0285
            ENDCASE RETURN (FALSE);                                  0286
         = parsehelp:                                                0287
            IF curptr.begnodeptr = curptr.curnodeptr THEN           01106
               *string* _ "NUM:";                                    0288
         = parseqmark:                                               0740
            BEGIN                                                    0741
            *string* _ "NUMBER";                                     0742
            RETURN;                                                  0743
            END;                                                     0744
      ENDCASE;                                                       0289
      RETURN (&curptr);                                              0290
      END.
                                                                     0291
                                                                     0292
% dnls or tnls %                                                    0948
   (isdnls) PROC(curptr, parsemode, string);                         0949
      % return TRUE if DNLS %                                       0950
      REF curptr, string;                                            0951
      %------------%                                                 0952
      CASE parsemode OF                                              0953
         = parsing:                                                  0954
            IF nlmode = typewriter THEN RETURN (FALSE);              0958
         ENDCASE;                                                    0966
      RETURN (&curptr);                                              0967
      END.
                                                                     0968
                                                                     0969
```

```
   (istnls) PROC(curptr, parsemode, string);                          0970
      % return TRUE if TNLS %                                         0971
      REF curptr, string;                                             0972
      %------------%                                                  0973
      CASE parsemode OF                                               0974
         = parsing:                                                   0975
            IF nlmode = fulldisplay THEN RETURN (FALSE);              0976
         ENDCASE;                                                     0977
      RETURN (&curptr);                                               0978
      END.                                                            0979
                                                                      0980
% RETURNS TRUE OR FALSE %                                            0941
   (false) PROC(result, parsemode);                                   0942
      IF parsemode = parsing THEN RETURN(0)                           0943
      ELSE RETURN(result);                                            0944
      END.                                                            0945
   (true) PROC(result, parsemode);                                    0946
      RETURN(result); END.                                            0947
%prompt for lsel, dsel, and ssel%                                     0812
   (prmptlsel) PROC(curptr, parsemode, string);                       0839
      % generate correct prompting and ? response for an LSEL
      (excluding the type of parameter to be selected). %             0840
      REF curptr, string;                                             0841
      %------------%                                                  0842
      CASE parsemode OF                                               0843
         = parsehelp:                                                 0849
            IF curptr.begnodeptr = curptr.curnodeptr THEN             0984
               BEGIN                                                  0912
               IF nlmode = fulldisplay THEN *string* _ "B/T"          0850
               ELSE *string* _ "T";                                   0851
               IF inprompts = partprompts THEN                        0907
                  *string* _ *string*, ":                             0908
               ELSE                                                   0909
                  *string* _ *string*, "/[A]:";                       0910
               END;                                                   0911
         = parseqmark:                                                0852
            BEGIN                                                     0853
            IF nlmode = fulldisplay THEN *string* _ "BUG", 0,
            "TYPEIN", 0, "OPTION ADDRESS", 0                          0854
            ELSE *string* _ "TYPEIN", 0, "OPTION ADDRESS", 0;         0855
            END;                                                      0857
         ENDCASE;                                                     0858
      RETURN (&curptr);                                               0859
      END.                                                            0860
                                                                      0861
   (prmptdsel) PROC(curptr, parsemode, string);                       0879
      % generate correct prompting and ? response for an DSEL
      (excluding the type of parameter to be selected). %             0880
      REF curptr, string;                                             0881
      %------------%                                                  0882
      CASE parsemode OF                                               0883
         = parsehelp:                                                 0884
```

```
            IF curptr.begnodeptr = curptr.curnodeptr THEN          01107
               IF nlmode = fulldisplay THEN *string* _ "B/A:"       0885
               ELSE *string* _ "A:";                                0886
         = parseqmark:                                              0887
            BEGIN                                                   0888
            IF nlmode = fulldisplay THEN *string* _ "BUG", 0,
            "ADDRESS", 0                                            0889
            ELSE *string* _ "ADDRESS", 0;                           0890
            END;                                                    0891
         ENDCASE;                                                   0892
      RETURN (&curptr);                                             0893
      END.
                                                                    0894
                                                                    0895
   (prmptssel) PROC(curptr, parsemode, string);                     0862
      % generate correct prompting and ? response for an SSEL
      (excluding the type of parameter to be selected). %           0863
      REF curptr, string;                                           0864
      %------------%                                               0865
      CASE parsemode OF                                             0866
         = parsehelp:                                               0867
            IF curptr.begnodeptr = curptr.curnodeptr THEN          01108
               BEGIN                                                0905
               IF nlmode = fulldisplay THEN *string* _ "B/A"        0868
               ELSE *string* _ "A";                                 0869
               IF inprompts = partprompts THEN                      0896
                  *string* _ *string*, ':                           0899
               ELSE                                                 0898
                  *string* _ *string*, "/[T]:";                     0897
               END;                                                 0906
         = parseqmark:                                              0870
            BEGIN                                                   0871
            IF nlmode = fulldisplay THEN *string* _ "BUG", 0,
            "ADDRESS", 0, "OPTION TYPEIN", 0                        0872
            ELSE *string* _ "ADDRESS", 0, "OPTION TYPEIN", 0;       0873
            END;                                                    0874
         ENDCASE;                                                   0875
      RETURN (&curptr);                                             0876
      END.
                                                                    0877
                                                                    0878
% BACKUP CONTROL FUNCTION %                                         0293
   (setbkup) PROCEDURE( % sets backup point for parse %             0294
      % FORMAL ARGUMENTS %                                         0295
         resultptr,     % ptr to result record %                   0296
         parsemode);     % parsing mode %                          0297
      REF resultptr;                                                0298
      %------------%                                               0299
      % this routine is a dummy and does nothing except mark the backup
      point for the command repeat function %                       0300
      RETURN( &resultptr );                                         0301
      END.
                                                                    0302
% KEYWORD SEQUENCE DATA STRUCTURE MANIPULATION ROUTINES %           0303
   (seqbinit) PROC( % initializes a keyword data structure %        0304
      % The sequence structure is allocated here and is initialized for
```

```
    subsequent store operations %                                          0305
    % FORMAL ARGUMENTS %                                                   0306
       resultptr,      % ptr to the result record %                       0307
       parsemode);     % parsing mode %                                   0308
    LOCAL addr;                                                           01159
    REF resultptr, addr;                                                   0309
    %------------%                                                         0310
    CASE parsemode OF                                                      0311
       = parsing:                                                          0312
          BEGIN                                                            0313
          resultptr[1] _ getstring(255, $dspblk ); % allocate 51 words
          %                                                                0314
          resultptr _ resultptr[1]+1; % set up address of block %          0315
          [resultptr] _ 0; % initialize count to zero %                    0316
          END;                                                             0317
       = backup,                                                           0318
       = cleanup:                                                          0319
          BEGIN                                                           01155
          %deallocate any strings allocated for the sequence block%
                                                                          01156
             FOR &addr _ resultptr[1] UP UNTIL >= resultptr[1] +51 DO
                                                                          01157
                IF addr.LH = 400B THEN freestring(addr.RH, $dspblk);
                                                                          01158
          % deallocate the sequence block %                              0320
             freestring( resultptr[1], $dspblk );                         0321
          END;                                                            01154
       ENDCASE;                                                           0322
    RETURN( &resultptr );                                                 0323
    END.
                                                                          0324
(seqbadd) PROC( % adds a keyword to a keyword data structure %            0325
    % FORMAL ARGUMENTS %                                                  0326
       resultptr,      % ptr to the result record %                       0327
       parsemode,      % parsing mode %                                   0328
       seqbstrptr,     % ptr to sequence structure %                      0329
       strptr);        % ptr to keyword string %                          0330
    LOCAL count, ptr, str;                                                0331
    REF resultptr, strptr, seqbstrptr, ptr, str;                          0332
    %------------%                                                         0333
    CASE parsemode OF                                                     0334
       = parsing:                                                         0335
          BEGIN                                                           0336
          &ptr _ seqbstrptr; % get ptr to structure %                     0337
          count _ ptr; % current count %                                  0338
          BUMP count;                                                     0339
          IF strptr.LH = 400B THEN %allocate and copy astring%            01160
             BEGIN                                                        01161
             &str _ strptr.RH;                                            01162
             ptr[count] _ getstring(str.L, $dspblk);                      01163
             *[ptr[count]]* _ *str*;                                      01167
             ptr[count].LH _ 400B;                                        01168
             END                                                          01165
          ELSE                                                            01166
             ptr[count] _ strptr;                                         0340
          ptr _ count;                                                    0341
```

```
            END;                                                    0342
         ENDCASE;                                                   0343
      RETURN( &resultptr );                                         0344
      END.
                                                                    0345
%abort command as though user typed CD%                             0765
   (abort)      %abort current command specification%
   PROCEDURE (resultptr, parsemode);                                0752
      %clear input buffer and simulate a CD%                        0753
      REF resultptr;                                                0754
      CASE parsemode OF                                             0755
         = parsing:                                                 0756
            BEGIN                                                   0757
            clrbuf(0);  %clear input buffer%                        0758
            curchr _ CD;                                            0759
            unput();                                                0760
            END;                                                    0761
         ENDCASE;                                                   0762
      RETURN( &resultptr );                                         0763
      END.
                                                                    0764
%cut back the pathstack%                                            0766
   (cutback)    %cut back the pathstack to the frame for setcutback%
   PROCEDURE (curptr, parsemode);                                   0767
      REF curptr;                                                   0769
      CASE parsemode OF                                             0770
         = parsing:                                                 0771
            SIGNAL(cutpathstk);                                     0801
         ENDCASE;                                                   0777
      RETURN( &curptr );                                            0778
      END.
                                                                    0779
   (setcutback) %set cutbackstop  to point to current frame%
   PROCEDURE (curptr, parsemode);                                   0802
      REF curptr;                                                   0804
      CASE parsemode OF                                             0805
         = parsing:                                                 0806
            cutstop _ &curptr;                                      0807
         ENDCASE cutstop _ 0;                                       0808
      RETURN( &curptr );                                            0809
      END.
                                                                    0810
% CLEAR NAME AREA %                                                 0346
   (clearname) PROC(curptr, parsemode, string);                     0347
      % clears the name area and returns TRUE %                     0348
      REF curptr, string;                                           0349
      %------------%                                               0350
      CASE parsemode OF                                             0351
         = parsing:                                                 0352
            dn ($"");                                               0353
         ENDCASE;                                                   0354
      RETURN (&curptr);                                             0355
      END.
                                                                    0356
                                                                    0357
% CLIST UTILITY %                                                  0358
```

```
(clist)    PROCEDURE (type, fno1, fno2); %build clist%                    0359
   %This routine constructs the correspondence list according to the
   parameters passed as follows: See clhdr and clistr for format of
   clist.                                                                 0360
      If either fno1 or fno2 are > 0 then only stid's                     0361
      belonging to those files are used.                                  0362
      if type = 0 then clhead.clcnt is set to zero.                       0363
      if type .A 1 = 1 then the csp's are used.                           0364
      if type .A 2 = 2 then the frozen list entries are                   0365
      used.                                                               0366
      if type .A 4 = 4 then the return ring entries are used.             0367
      if type .A 8 = 8 then the markers are used.                         0368
      any combination of the above is valid.%                             0369
   %----------------%                                                     0370
   LOCAL                                                                  0371
      end1, entry1, cl, fz, hd, mk, last1, a, b, srr, frr, i, j, fn,
      cb, bump;                                                           0372
   LOCAL STRING fnam1[150], fnam2[150];                                   0373
   REF entry1, cl, fz, mk, srr, frr, fn, bump;                            0374
   clhead.clbuff _ &cl _ clistb;                                          0375
   clhead.clcnt _ 0;                                                      0376
   clhead.clfno1 _ fno1;                                                  0377
   clhead.clfno2 _ fno2;                                                  0378
   clhead.cltype _ type;                                                  0379
   IF type = 0 THEN RETURN;                                               0380
   IF type .A 4 THEN %file return ring%                                   0381
      BEGIN                                                               0382
      %do not trust file numbers--use file name%                          0383
      *fnam1* _ NULL;                                                     0384
      *fnam2* _ NULL;                                                     0385
      IF fno1 THEN filnam(fno1, $fnam1);                                  0386
      IF fno2 THEN filnam(fno2, $fnam2);                                  0387
      END;                                                                0388
   end1 _ (&entry1 _ $dpyarea) + dacnt*dal;                               0389
   ON SIGNAL                                                             01121
      = clisterr:    %signalled only by upclptr%                         01122
         BEGIN                                                          01123
         cb _ getblk(clistsz+100, $dspblk) + bhl;                       01124
            %allocate a bigger block for the correspondence list%
                                                                        01134
         mvbfbf(clistb, cb, clistsz);   %move old block to new%         01126
         &cl _ cb + (&cl - clistb);     %cl to point into new block%
                                                                        01127
         freeblk(clistb - bhl, $dspblk);  %free old buffer%             01128
         clistb _ clhead.clbuff _ cb;  %point to new block%             01129
         clistsz _ clistsz + 100;       %update size%                   01130
         GOTO bump;                     %address of label to go to%
                                                                        01131
         END;                                                           01132
      ELSE;          %ignore other signals%                             01133
   UNTIL &entry1 >= end1 DO                                               0390
      BEGIN                                                               0391
      IF type .A 1 THEN %csp's%                                           0392
         BEGIN                                                            0393
         &bump _ $bumpcsp;                                              01138
         IF entry1.daexis AND NOT entry1.daempty THEN                     0394
```

```
        IF (fno1 = 0 AND fno2 = 0) OR
         (entry1.dacsp.stfile = fno1) OR
         (entry1.dacsp.stfile = fno2) THEN                          0395
           BEGIN                                                    0396
           cl.clst1 _ entry1.dacsp;                                 0397
           cl.clcc1 _ entry1.daccnt;                                0398
           cl.clst2 _ endfil;                                       0399
           cl.clcc2 _ 1;                                            0400
           cl.clfixed _ FALSE;                                     01111
           IF cl.clst1 = curmkr THEN                               01115
              cl.clcurmkr _ TRUE  %for later update of curmkr%
                                                                   01116
           ELSE cl.clcurmkr _ FALSE;                               01117
           upclptr($&cl);                                           0401
        (bumpcsp):                                                 01142
           BUMP clhead.clcnt;                                       0402
           END                                                      0403
        ELSE NULL;                                                  0404
     END;                                                           0405
  IF type .A 2 THEN %frozen list%                                   0406
     BEGIN                                                          0407
     &bump _ $bumpfzl;                                             01137
     IF entry1.daexis AND NOT entry1.daempty AND &fz _              0408
     entry1.dafrzl THEN                                             0409
        DO                                                          0410
           IF (fno1 = 0 AND fno2 = 0) OR
            (fz.fzstid.stfile = fno1) OR
            (fz.fzstid.stfile = fno2) THEN                          0411
              BEGIN                                                 0412
              cl.clst1 _ fz.fzstid;                                 0413
              cl.clcc1 _ cl.clcc2 _ 1;                              0414
              cl.clst2 _ endfil;                                    0415
              cl.clfixed _ TRUE;                                   01112
              upclptr($&cl);                                        0416
           (bumpfzl):                                              01139
              BUMP clhead.clcnt;                                    0417
              END                                                   0418
           ELSE NULL                                                0419
        UNTIL (&fz _ fz.fznext) = 0;                                0420
     END;                                                           0421
  IF type .A 4 THEN %file return ring%                              0422
     BEGIN                                                          0423
     &bump _ $bumpfrr;                                             01136
     IF entry1.daexis THEN                                          0424
        BEGIN                                                       0425
        IF &frr _ entry1.dalink THEN                                0426
           FOR i _ frrlength(&frr) DOWN UNTIL < 0 DO                0427
              BEGIN                                                 0428
              ON SIGNAL ELSE REPEAT LOOP; %in case top entry
              non-existant%                                        01150
              &fn _ readfrring(&frr, i : &srr);                     0429
              ON SIGNAL ELSE;                                      01151
              IF (fno1 = 0 AND fno2 = 0)
              OR (fno1 AND a _ (*fn* = *fnam1*))
              OR (fno2 AND b _ (*fn* = *fnam2*)) THEN               0430
                 BEGIN                                              0431
```

```
                    FOR j _ srrlength(&srr) DOWN UNTIL < 0 DO    0432
                       BEGIN                                     0433
                       cl.clst1 _ readsrring(&srr, j : cl.clcc1);
                                                                 0434
                       cl.clst1.stfile _                         0435
                          IF a THEN fno1 ELSE fno2;              0436
                       cl.clst2 _ endfil;                        0437
                       cl.clcc2 _ 1;                             0438
                       cl.clfixed _ FALSE;                      01113
                       upclptr($&cl);                            0439
                    (bumpfrr):                                  01140
                       BUMP clhead.clcnt;                        0440
                       END;                                      0441
                    END;                                         0442
                 END                                             0443
           ELSE err($"no file return ring in clist");            0444
           END;                                                  0445
        END;                                                     0446
     IF type .A 8 THEN %markers%                                 0447
        BEGIN                                                    0448
        &bump _ $bumpmkr;                                       01135
        IF entry1.daexis AND NOT entry1.daempty THEN             0449
           IF  (fno1 = 0 AND fno2 = 0) OR
           (entry1.dacsp.stfile = fno1) OR                       0450
           (entry1.dacsp.stfile = fno2) THEN                     0451
              BEGIN                                              0452
              &mk _ $mkrtb - $filhed + (hd _                     0453
                 filhdr(entry1.dacsp.stfile));                   0454
              last1 _ &mk + [$mkrtbl - $filhed + hd] * mkrl;     0455
              %bounds check%                                     0456
                 IF (last1 - &mk) > [$mkrtxn + hd - $filhed] THEN
                                                                 0457
                    BEGIN                                        0458
                    [$mkrtbl - $filhed + hd] _ [$mkrtxn + hd -
                    $filhed] / mkrl;                             0459
                    last1 _ &mk + [$mkrtbl - $filhed + hd] * mkrl;
                                                                 0460
                    END;                                         0461
              UNTIL &mk >= last1 DO                              0462
                 BEGIN                                           0463
                 cl.clst1 _ 0;                                   0464
                 cl.clst1.stfile _ entry1.dacsp.stfile;          0465
                 cl.clst1.stpsid _ mk.mkpsid;                    0466
                 cl.clcc1 _ mk.mkccnt;                           0467
                 cl.clst2 _ endfil;                              0468
                 cl.clcc2 _ 1;                                   0469
                 cl.clfixed _ FALSE;                            01114
                 &mk _ &mk + mkrl;                               0472
                 upclptr($&cl);                                  0470
              (bumpmkr):                                        01141
                 BUMP clhead.clcnt;                              0471
                 END;                                            0473
              END                                                0474
           ELSE NULL;                                            0475
        END;                                                     0476
     &entry1 _ &entry1 + dal;                                    0477
```

```
      END;                                                          0478
   RETURN;                                                          0480
   END.
                                                                    0481
(upclptr) PROCEDURE (claddr);                                       01143
   REF claddr;                                                      01145
   claddr _ claddr + cll;  %increment to point to next record slot%
                                                                    01146
   IF claddr > clistb + clistsz - cll THEN                          01147
      SIGNAL(clisterr, $"correspondence list block too small");     01148
   RETURN;                                                          01149
   END.
                                                                    01144
(clupdt)   PROCEDURE;  %unbuild clist%                              0482
   %This routine updates various things from the correspondence list
   according to the parameters in the clist header (clhead):        0483
      If either clhead.clfno1 or clhead.clfno2 are > 0 then         0484
      only stid's belonging to those files are updated.             0485
      if cltype = 0 then none of the following is changed.          0486
      if cltype .A 1 = 1 then the csp's are updated.                0487
      if cltype .A 2 = 2 then the frozen lists are updated.         0488
      if cltype .A 4 = 4 then the return ring is updated.           0489
      if cltype .A 8 = 8 then the markers are updated.              0490
      any combination of the above is valid.                        0491
   See clhdr and clistr for format of clist.%                       0492
      % if clst1 = endfil, then this statement has been deleted from
      the file and clst2 points to the "next" statement in the file.
      if stcc1 changes then the text that used to be pointed to has
      been deleted or moved and clcc1 has been updated to something
      reasonable.%                                                  0914
   %----------------%                                              0493
   LOCAL                                                            0494
      end1, stid, entry1, cl, fz, fz2, hd, view1, view2,            0495
      mklngth, mkold, mk, last1, srr, frr, fn, i, j, cc, a, b;      0496
   LOCAL TEXT POINTER b1;                                           0913
   LOCAL STRING fnam1[150], fnam2[150];                             0497
   REF entry1, cl, fz, fz2, mklngth, mk, srr, frr, fn;              0498
   IF clhead.cltype = 0 THEN RETURN;                                0499
   &cl _ clhead.clbuff;                                             0500
   IF clhead.cltype .A 4 THEN %link1 ring%                          0501
      BEGIN                                                         0502
      %do not trust file numbers--use file name%                    0503
      *fnam1* _ NULL;                                               0504
      *fnam2* _ NULL;                                               0505
      IF clhead.clfno1 THEN filnam(clhead.clfno1, $fnam1);          0506
      IF clhead.clfno2 THEN filnam(clhead.clfno2, $fnam2);          0507
      END;                                                          0508
   end1 _ (&entry1 _ $dpyarea) + dacnt*dal;                         0509
   UNTIL &entry1 = end1 DO                                          0510
      BEGIN                                                         0511
      IF clhead.cltype .A 1 THEN %csp's%                            0512
         BEGIN                                                      0513
         IF entry1.daexis AND NOT entry1.daempty THEN               0514
            IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
            (clhead.clfno1 AND entry1.dacsp.stfile = clhead.clfno1)
            OR
```

```
        (clhead.clfno2 AND entry1.dacsp.stfile = clhead.clfno2)
        THEN                                                    0515
           BEGIN                                                0516
           IF cl.clst1 = endfil OR cl.clst1.stfile NOT=         0517
           entry1.dacsp.stfile THEN %replace by clst2%          0518
              BEGIN                                             0519
              IF cl.clst2 = endfil THEN                         0915
                 BEGIN                                          0916
                 entry1.dacsp _ endfil;                         0917
                 entry1.daccnt _ 1;                             0918
                 entry1.daempty _ TRUE;                         0919
                 END                                            0920
              ELSE                                              0921
                 BEGIN                                          0922
                 b1 _ cl.clst2;                                 0520
                 b1[1] _ cl.clcc2;                              0521
                 IF POS b1 >= SE(b1) THEN FIND SE(b1) CH ^b1;
                                                                0522
                 entry1.dacsp _ b1;                             0523
                 entry1.daccnt _ b1[1];                         0524
                 END;                                           0923
              IF cl.clcurmkr THEN FIND b1 ^curmkr;             01118
                 %curmkr needs to be updated, too%             01119
              END                                               0525
           ELSE                                                 0526
              BEGIN                                             0527
              IF entry1.dacsp NOT= cl.clst1 THEN                0528
                 entry1.dacsp _ cl.clst1;                       0924
              IF entry1.daccnt NOT= cl.clcc1 THEN               0529
                 entry1.daccnt _ cl.clcc1;                      0925
              END;                                              0530
           &cl _ &cl + cll;                                     0531
           END;                                                 0532
     END;                                                       0533
  IF clhead.cltype .A 2 THEN %frozen list%                      0534
     BEGIN                                                      0535
     IF entry1.daexis AND NOT entry1.daempty AND &fz _          0536
     entry1.dafrzl THEN                                         0537
        DO                                                      0538
           IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
           (clhead.clfno1 AND fz.fzstid.stfile = clhead.clfno1)
           OR
           (clhead.clfno2 AND fz.fzstid.stfile = clhead.clfno2)
           THEN                                                 0539
              BEGIN                                             0540
              IF cl.clst1 = endfil THEN                         0541
                 BEGIN %statement was deleted%                  0542
                 IF (&fz2 _ entry1.dafrzl) = &fz THEN           0543
                    entry1.dafrzl _ fz.fznext                   0544
                 ELSE                                           0545
                    DO IF fz2.fznext = &fz THEN                 0546
                       BEGIN                                    0547
                       fz2.fznext _ fz.fznext;                  0548
                       EXIT;                                    0549
                       END                                      0550
                    UNTIL (&fz2 _ fz2.fznext) = 0;              0551
```

```
                    &fz2 _ fz.fznext;                                   0926
                    fz.fznext _ fzfree := &fz;                          0552
                    IF &fz2 = 0 THEN EXIT LOOP;                         0928
                    END                                                 0553
                 ELSE                                                   0554
                    IF cl.clst1 # fz.fzstid THEN                        0555
                       fz.fzstid _ cl.clst1;                            0556
                 &cl _ &cl + cll;                                       0557
                 END                                                    0558
           UNTIL (&fz _ fz.fznext) = 0;                                 0559
        END;                                                            0560
     IF clhead.cltype .A 4 THEN %file return ring%                      0561
        BEGIN                                                           0562
        IF entry1.daexis THEN                                           0563
           BEGIN %cycle through frr and srr's%                          0564
           IF &frr _ entry1.dalink THEN                                 0565
              FOR i _ frrlength(&frr) DOWN UNTIL < 0 DO                 0566
                 BEGIN                                                  0567
                 ON SIGNAL ELSE REPEAT LOOP; %in case top entry
                 non-existant%                                         01152
                 &fn _ readfrring(&frr, i : &srr);                      0568
                 ON SIGNAL ELSE;                                       01153
                 IF (clhead.clfno1 = 0 AND clhead.clfno2 = 0)
                 OR (clhead.clfno1 AND a _ (*fn* = *fnam1*))
                 OR (clhead.clfno2 AND b _ (*fn* = *fnam2*)) THEN
                                                                        0569
                    BEGIN %found a match -- update stid, cc for each
                    entry in srr%                                       0570
                    FOR j _ srrlength(&srr) DOWN UNTIL < 0 DO           0571
                       BEGIN                                            0572
                       stid _ readsrring(&srr, j : cc, view1,
                       view2);                                          0573
                       IF cl.clst1 NOT= endfil THEN                     0574
                          BEGIN                                         0575
                          IF stid.stpsid NOT= cl.clst1.stpsid OR cc
                          NOT= cl.clcc1 THEN                            0576
                             storesrring(&srr, j, cl.clst1,
                             cl.clcc1, view1, view2)                    0577
                          END                                           0578
                       ELSE                                             0579
                          storesrring(&srr, j, cl.clst2, cl.clcc2,
                          view1, view2);                                0580
                       &cl _ &cl + cll;                                 0581
                       END;                                             0582
                    END;                                                0583
                 END                                                    0584
           ELSE err($"no file return ring in clupdt");                  0585
           END;                                                         0586
        END;                                                            0587
     IF clhead.cltype .A 8 THEN %markers%                               0588
        BEGIN                                                           0589
        IF entry1.daexis AND NOT entry1.daempty THEN                    0590
           IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
           (clhead.clfno1 AND entry1.dacsp.stfile= clhead.clfno1) OR
           (clhead.clfno2 AND entry1.dacsp.stfile= clhead.clfno2)
           THEN                                                         0591
```

```
            BEGIN                                                   0592
            &mk _ mkold _ $mkrtb - $filhed + (hd _                  0593
               filhdr(clhead.clfno1));                              0594
            last1 _ &mk + [&mklngth _ $mkrtbl - $filhed             0595
            + hdJ*mkrl;                                             0596
            UNTIL &mk = last1 DO                                    0597
               BEGIN                                                0598
               IF cl.clst1 = endfil THEN %delete marker%            0607
                  BEGIN                                             0608
                  mvbfbf(&mk + mkrl, &mk,                           0609
                  last1-&mk-mkrl);                                  0610
                  BUMP DOWN mklngth;                                0611
                  last1 _ last1 - mkrl;                             0612
                  &cl _ &cl + cll;                                  0933
                  END                                               0613
               ELSE                                                 0614
                  BEGIN                                             0929
                  IF mk.mkpsid NOT= cl.clst1.stpsid THEN            0931
                     mk.mkpsid _ cl.clst1.stpsid;                   0599
                  IF mk.mkccnt NOT= cl.clcc1 THEN                   0932
                     mk.mkccnt _ cl.clcc1;                          0601
                  &cl _ &cl + cll;                                  0602
                  &mk _ &mk + mkrl;                                 0603
                  END;                                              0930
               END;                                                 0604
            END;                                                    0615
         END;                                                       0616
      &entry1 _ &entry1 + dal;                                      0617
      END;                                                          0618
   RETURN;                                                          0619
   END.
                                                                    0620
(dpset) PROCEDURE (option, stid1, stid2, stopstid);                 0621
   %set global variables for recreate display routines.  In some
   cases, stid1 and stid2 are passed merely to indicate file
   involvement, not because they will be reformatted.%              0622
   cdtype _ option;                                                 0623
   IF stid1 = stid2 THEN stid2 _ endfil;                            0657
   (cdstd1, cdstd2) _ (stid1, stid2);                               0624
   IF option = dsprfmt THEN %save statements before edit%           0656
      BEGIN                                                         0658
      IF stid1 NOT= endfil THEN                                     0662
         *cdstr1* _ $SF(stid1) SE(stid1);                           0659
      IF stid2 NOT= endfil THEN                                     0663
         *cdstr2* _ $SF(stid2) SE(stid2);                           0661
      END;                                                          0660
   IF stopstid.stpsid = orgstid THEN cdstop _ endfil                0625
   ELSE cdstop _ stopstid;                                          0626
   RETURN END.                                                      0627
                                                                    0628
(dpstp) PROCEDURE (stid);                                           0629
   %called by text editing control routines to accept the stid of a
   bugged statement as inpt and return an appropriate stid for the
   display parameter CDSTOP (an stid on the same level or higher
   which appears after 'stid' in the file).%                        0630
   UNTIL NOT <FILMNP, getftl> (stid)                                0631
```

```
      DO stid _ <FILMNP, getsuc> (stid);                                0632
    RETURN (stid _ getsuc (stid));                                      0633
    END.                                                                0634
                                                                        0635
% NUMBER CONVERSION %                                                  0636
  (getpint)  % convert 2 text pointers to integer %                     0637
    PROCEDURE                                                           0638
      (tp1,  % starting text pointer %                                  0639
      tp2 ); % ending text pointer %                                    0640
    LOCAL STRING                                                        0641
      locstr[50];    % temp string %                                   0642
    REF tp1, tp2;                                                       0643
                                                                        0644
    *locstr* _ tp1 tp2;                                                 0645
    FIND SF(*locstr*);                                                  0646
    CASE READC OF                                                       0647
      = D : REPEAT CASE;                                                0648
      = ENDCHR : EXIT CASE;                                             0649
      ENDCASE err( $"Illegal Number" );                                 0650
    RETURN(VALUE($locstr));                                             0651
    END.                                                                0652
% give warning message if experimental system %                         0664
  (xwarning) PROC(curptr, parsemode);                                   0665
    % display a warning message to the user if he is using the
    experimental system and returns TRUE %                              0666
    REF curptr;                                                         0667
    %------------%                                                      0668
    CASE parsemode OF                                                   0669
      = parsing:                                                        0670
        IF jdebug THEN                                                  0671
          dismes (2, $"WARNING: EXPERIMENTAL SYSTEM, use at your
          own risk!");                                                  0676
      ENDCASE;                                                          0672
    RETURN (&curptr);                                                   0673
    END.
                                                                        0674
                                                                        0675
FINISH                                                                  0653
```

PSUSERUP

```
< NLS, PSUSEROP.NLS.23, >, 6-Dec-77 13:08 JCP ;;;;
FILE psuserop % L10 <rel-nls>psuserop %% (L10,) (rel-nls,psuserop.rel,)
%                                                                    02
% USER-OPTIONS SUBSYSTEM %                                           03
   % "X" level support routines %                                    04
      % NOTE: These routines all make use of (nls,userdata,) %       05
      (xuoinit) % initialize user-options subsystem %                06
         PROCEDURE (resultptr, parsemode);                           07
         LOCAL STRING filmns[40];                                    08
         REF resultptr;                                              09
         CASE parsemode OF                                           010
            = parsing:                                               011
               BEGIN                                                0798
               IF NOT uojfn THEN                                    0796
                  IF NOT uoget() OR NOT uojfn THEN                  0802
                     err($"you can not modify your PROFILE at this
                     time");                                        0797
               % map page into users address space at "userdata" %  0918
                  r1.LH _ uojfn;        % profile file jfn %        0919
                  r1.RH _ 0;            % page zero %               0920
                  r2.LH _ 400000B;      % this fork %               0921
                  r2.RH _ $userdata / 1000B;                        0922
                  r3 _ racc .V wacc;    %read and write access %    0923
                  !pmap();                                          0924
               % initialize if nessecary %                         0925
                  IF NOT userdata THEN                              0926
                     BEGIN                                          0927
                     uoreset();  % initialize to system defaults %  0928
                     userdata _ TRUE;  % say initialized %          0929
                     END;                                           0930
               % set recognition mode if this is within a process
               command; force demand mode %                        02323
                  IF auxinput THEN                                  02324
                     BEGIN                                          02325
                     recogmode _ mdemand;                           02326
                     recog2mode _ mdemand;                          02327
                     END;                                           02328
               RETURN(TRUE);                                        0931
               END;                                                 0800
            ENDCASE;                                                014
         RETURN( &resultptr );                                      015
         END.                                                       016
      (xuoterm) % terminate user-options subsystem %                017
         PROCEDURE (resultptr, parsemode);                          018
         REF resultptr;                                             019
         CASE parsemode OF                                          020
            = parsing:                                              021
               BEGIN                                                02339
               IF auxinput THEN                                     02330
                  BEGIN %set back to user's own ones while we map out
                  the page%                                         02331
                  recogmode _ auxmod;                               02332
                  recog2mode _ auxmd2;                              02333
                  END;                                              02334
               % set page access back to read and copy on write %   022
               uoaccess($userdata / 1000B, racc .V cwacc);          023
```

```
            IF auxinput THEN                                          02335
               BEGIN %put it back to demand mode while we continue to
               process commands%                                      02336
               recogmode _ recog2mode _ mdemand;                      02337
               END;                                                   02338
            END;                                                      02340
         ENDCASE;                                                       024
      RETURN( &resultptr );                                             025
      END.                                                              026
   (xuocontchars)    % set control characters for specified device %
                                                                        027
      PROCEDURE (resultptr, parsemode, dev, cc, char, echo);            028
      LOCAL i, j, k, tp,  devcode, cchar, echochar, chars[10];          029
      LOCAL STRING temp[20];                                            030
      REF resultptr, dev, cc, char, echo, tp;                           031
      CASE parsemode OF                                                 032
         = parsing:                                                     033
            BEGIN                                                       034
            % convert device string into internal code %                035
               devcode _ strdev(&dev);                                  036
            % convert ctrlchar string into internal code %              037
               cchar _ strchr(&cc);                                     038
            % extract control characters from string %                  039
               &tp _ &char+d2sel;                                       040
               *temp* _ char tp;                                        041
               CCPOS SF(*temp*);                                        042
               FOR j _ 0 UP UNTIL = 9 DO                                043
                  CASE chars[j] _ READC OF                              044
                     = SP ,                                             045
                     = ', : REPEAT CASE ;                               046
                     = CA, = CD: err($" you can not redefine <CA> or
                     <CD>");                                            047
                     IN ['A,'Z] : err($"invalid character
                     specified");                                       048
                     IN ['a,'z] : err($"invalid character
                     specified");                                       049
                     IN ['0,'9] : err($"invalid character
                     specified");                                       050
                     = ENDCHR : EXIT;                                   051
                     ENDCASE NULL;                                      052
            % get echo character from string %                          053
               &tp _ &echo+d2sel;                                       054
               *temp* _ echo tp;                                        055
               CCPOS SF(*temp*);                                        056
               IF FIND "null" / "NULL"  THEN echochar _ nullch          057
               ELSE                                                     058
                  IF (echochar _ READC) = ENDCHR                        059
                     THEN echochar _ nullch;                            060
            % undo current settings for "cc" (this device) %            061
               FOR i _ 0 UP UNTIL = 100 DO                              062
                  BEGIN                                                 063
                  IF cctbl[i].ccdevice = devcode AND
                  cctbl[i].cctype = cchar THEN                          064
                     cctbl[i] _ 0;                                      065
                  END;                                                  066
            % undo current settings for "chars" (this device) %        067
```

```
            FOR i _ 0 UP UNTIL = 100 DO                                  068
               FOR k _ 0 UP UNTIL = j DO                                 069
                  BEGIN                                                  070
                  IF cctbl[i].ccdevice = devcode AND
                  cctbl[i].ccchar = chars[k] THEN                        071
                     cctbl[i] _ 0;                                       072
                  END;                                                   073
            % now setup new user-options stuff for "cc" %                074
               FOR k _ 0 UP UNTIL = j DO                                 075
                  BEGIN                                                  076
                  i _ uotblget();                                        077
                  cctbl[i].ccchar _ chars[k];                            078
                  cctbl[i].ccdevice _ devcode;                           079
                  cctbl[i].cctype _ cchar;                               080
                  cctbl[i].ccecho _ echochar;                            081
                  END;                                                   082
            % setup translate and "break" tables %                       083
               initch(nldevice);     % do translate tables %            084
               initbtbl();           % do the break table %             085
            END;                                                         086
         ENDCASE;                                                        087
      RETURN( &resultptr );                                              088
      END.
                                                                         089
   (xuocurcon)       % set current context length %                      090
      PROCEDURE (resultptr, parsemode, numbr);                           091
      REF resultptr, numbr;                                              092
      CASE parsemode OF                                                  093
         = parsing:                                                      094
            tslshchars _ getpint(&numbr, &numbr+d2sel);                  095
         ENDCASE;                                                        096
      RETURN( &resultptr );                                              097
      END.
                                                                         098
   (xuodisplay)      % set display parameters %                        02310
      PROCEDURE (resultptr, parsemode, ptype, param);                  02311
      REF resultptr, ptype, param;                                     02312
      CASE parsemode OF                                                02313
         = parsing:                                                    02314
            CASE ptype OF                                              02315
               = 135 %- right -%:    udpcolmax _ getpint(&param,
               &param+d2sel);                                          02316
               = 211 %- wraparound -%:       udpwrapcol _
               getpint(&param, &param+d2sel);                          02317
               ENDCASE typeas($"invalid option specified");            02318
         ENDCASE;                                                      02319
      RETURN( &resultptr );                                            02320
      END.
                                                                       02321
   (xuoexclude)      % exclude a subsytem or user program to load at
   startup time%                                                       01553
      PROCEDURE ( resultptr, parsemode, param, param2);                01554
      LOCAL subptr,numwsubsys;                                         01555
      LOCAL STRING locstr[200];                                        01556
      REF resultptr, param,  param2,  subptr ;                         01557
      CASE parsemode OF                                                01558
```

```
      = parsing:                                                    01559
         BEGIN                                                      01560
         CASE param2 OF                                             01561
            = 180 %- program -%: uoincnam(&param,$locstr);          01562
            = 176 %- subsystem -%: *locstr* _ *[param]*;            01563
         ENDCASE                                                    01564
            BEGIN                                                   01565
            dismes(1,$"illegal param2 passed to xuoinclude");
                                                                    01566
            RETURN(&resultptr);                                     01567
            END;                                                    01568
         %check if it is already included%                          01569
            numwsubsys _ ((usys1.M + 4)/ 5) + 1;                    01570
            FOR &subptr _ $usys1 UP numwsubsys UNTIL >$usys15 DO
                                                                    01571
               IF *subptr* = *locstr* THEN                          01572
                  BEGIN                                             01573
                  subptr.L _ 0;                                     01574
                  RETURN(&resultptr);                               01575
                  END;                                              01576
         %error if not included%                                    01577
            *locstr* _ *locstr* , " not currently included";        01578
            dismes(1,$locstr);                                      01579
         END;                                                       01580
      ENDCASE;                                                      01581
   RETURN(&resultptr);                                              01582
   END.                                                             01583
(xuofeedback)     % set feedback mode, etc %                        099
   PROCEDURE (resultptr, parsemode, type, param);                   0100
   LOCAL oldfbk;                                                    0812
   REF resultptr, type, param;                                      0101
   CASE parsemode OF                                                0102
      = parsing:                                                    0103
         CASE type OF                                               0104
            = 91 %- verbose -%:   fbackmode _ verbsmode;            0105
            = 181 %- terse -%:    fbackmode _ tersemode;            0106
            = 85 %- length -%:                                      0107
               BEGIN                                                0806
               oldfbk _ feedbk;                                     0811
               feedbk _ getpint(&param, &param+d2sel);              0108
               IF NOT feedbk > 0 THEN                               0805
                  BEGIN                                             0808
                  feedbk _ oldfbk;                                  0810
                  typeas($"invalid value for feedback specified");
                                                                    0813
                  END;                                              0809
               END;                                                 0807
            = 182 %- indenting -%:                                  0109
               fedind _ getpint(&param, &param+d2sel);              0110
            ENDCASE typeas($"invalid option specified");            0111
      = cleanup: fbctl(clearcfl);                                   0112
      ENDCASE;                                                      0113
   RETURN( &resultptr );                                            0114
   END.
                                                                    0115
(xuohearald)      % set herald %                                    0116
```

```
   PROCEDURE (resultptr, parsemode, type, param);                    0117
   REF resultptr, type, param;                                       0118
   CASE parsemode OF                                                 0119
      = parsing:                                                     0120
         BEGIN                                                       0121
         CASE type OF                                                0122
            = 91 %- verbose -%:   hrldmode _ multchar;               0123
            = 181 %- terse -%:    hrldmode _ onechar;                0124
            = 85 %- length -%:                                       0125
               hrldsize _ MAX( 1,                                    0126
                  MIN(hrldstr.M, getpint(&param, &param+d2sel)));
                                                                     0127
            ENDCASE typeas($"invalid option specified");             0128
         sethrld($ssysname);                                         0129
         END;                                                        0130
      ENDCASE;                                                       0131
   RETURN( &resultptr );                                             0132
   END.
                                                                     0133
(xuoinclude)      % include a subsytem or user program to load at
startup time%                                                        01509
   PROCEDURE ( resultptr, parsemode, param, param1, param2);         01510
   LOCAL subptr,numwsubsys,freeaddr;                                 01511
   LOCAL STRING errstr[200], locstr[200];                            01512
   REF resultptr, param, param1, param2,  subptr ;                   01513
   CASE parsemode OF                                                 01514
      = parsing:                                                     01515
         BEGIN                                                       01516
         CASE param2 OF                                              01517
            = 180 %- program -%: uoincnam(&param,$locstr);           01518
            = 176 %- subsystem -%: *locstr* _ *[param]*;             01519
         ENDCASE                                                     01520
            BEGIN                                                    01521
            dismes(2,$"illegal param2 passed to xuoinclude");
                                                                     01522
            RETURN(&resultptr);                                      01523
            END;                                                     01524
         CASE param1 OF                                              01525
            = 183 %- universal -%:                                   01526
               *usys1* _ *locstr*;                                   01527
            = 184 %- entry -%:                                       01528
               *usys2* _ *locstr*;                                   01529
            = 185 %- include -%:                                     01530
               BEGIN                                                 01531
               %check if it is already included%                     01532
                  numwsubsys _ ((usys1.M + 4)/ 5) + 1;               01533
                  FOR &subptr _ $usys1 UP numwsubsys UNTIL
                  >$usys15 DO                                        01534
                     IF *subptr* = *locstr* THEN
                     RETURN(&resultptr);                             01535
               %find first free entry in subsys list%                01536
                  freeaddr _ 0;                                      01537
                  &subptr _ $usys3; %usys1 and usys2 are reserved
                  for supervisor and entry usystems%                 01538
                  UNTIL freeaddr OR &subptr > $usys15 DO             01539
                     IF subptr.L = 0 THEN freeaddr _ &subptr ELSE
```

```
                    &subptr _ &subptr + numwsubsys;                    01540
                  IF &subptr > $usys15 THEN err($"no more entries
                  allowed, exclude one and try again");                01541
                  *[freeaddr]* _ *locstr*;                             01542
               END;                                                    01543
         ENDCASE                                                       01544
            BEGIN                                                      01545
            dismes(2,$"illegal param1 passed to xuoinclude");
                                                                       01546
            RETURN(&resultptr);                                        01547
            END;                                                       01548
         END;                                                          01549
      ENDCASE;                                                         01550
   RETURN( &resultptr);                                                01551
   END.                                                                01552
% not called                                                           01409
(xuolevel)        %% set ask for level adjust ON/OFF %%                01410
   PROCEDURE ( resultptr, parsemode, param);                           01411
   REF resultptr, param ;                                              01412
   CASE parsemode OF                                                   01413
      = parsing:                                                       01414
         CASE param OF                                                 01415
            = 1 : nolevadj _ FALSE;                                    01416
            = 2 : nolevadj _ TRUE;                                     01417
            ENDCASE typeas($"invalid option specified")                01418
      ENDCASE;                                                         01419
   RETURN( &resultptr);                                                01420
   END.                                                                01421
                                                                       01422
%                                                                      01423
(xuonamed)        %set default name delimiters%                        01366
   PROCEDURE ( resultptr, parsemode, param, param2);                   01367
   REF resultptr, param, param2 ;                                      01368
   CASE parsemode OF                                                   01369
      = parsing:                                                       01370
         BEGIN                                                         01405
         IF param THEN                                                 01378
            BEGIN                                                      01379
            CCPOS param; param _ READC;                                01380
            IF param = ENDCHR THEN param _ 0;                          01381
            dfnmdl _ param;                                            01382
            END;                                                       01388
         IF param2 THEN                                                01383
            BEGIN                                                      01384
            CCPOS param2; param2 _ READC;                              01385
            IF param2 = ENDCHR THEN param2 _ 0;                        01386
            dfnmdr _ param2;                                           01387
            END;                                                       01389
         END;                                                          01406
      ENDCASE;                                                         01375
   RETURN( &resultptr);                                                01376
   END.                                                                01377
(xuoextn) % set external names link file address %                      0948
   PROCEDURE ( resultptr, parsemode, param);                           0949
   LOCAL TEXT POINTER tp1;                                             0950
   LOCAL adstr[40];                                                    0951
```

```
    LOCAL STRING locstr[200];                                          0952
    REF resultptr, param ;                                             0953
    CASE parsemode OF                                                  0954
       = parsing:                                                      0955
          BEGIN                                                        0956
          IF param.stfile THEN                                         0957
             BEGIN                                                     0958
             lnkprs( &param, $adstr);                                  0959
             param _ adstr[ls];                                        0960
                param[1] _ adstr[ls+1];                                0961
             tp1 _ adstr[le];                                          0962
                tp1[1] _ adstr[le+1];                                  0963
             END                                                       0964
          ELSE                                                         0984
             BEGIN                                                     0985
             IF NOT FIND SF(param) ('(/'</"--") THEN                   0986
                ST param _ "< ", SF(param) SE(param);                  0987
             IF NOT FIND SE(param) (')/'>) THEN                        0988
                ST param _ SF(param) SE(param), " >";                  0989
             FIND SE(param) ^tp1;                                      0990
             END;                                                      0991
          %initialize string first time%                               0967
             IF NOT enlfstr.M THEN enlfstr _ defenlf;                  0968
          *locstr* _ *enlfstr*;                                        0969
          ON SIGNAL ELSE                                               0970
             BEGIN                                                     0971
             ON SIGNAL ELSE;                                           0972
             *enlfstr* _ *locstr*;                                     0973
             *locstr* _ "address must be less than ",                  0974
                STRING(enlfstr.M), "characters long";                  0975
             err( $locstr);                                            0976
             END;                                                      0977
          *enlfstr* _ param tp1;                                       0978
          END;                                                         0979
       ENDCASE;                                                        0980
    RETURN( &resultptr);                                               0981
    END.                                                               0982
                                                                       0983
(xuooutput)       % set output parameters %                           01660
    PROCEDURE (resultptr, parsemode, optype, ptype, param);            01661
    REF resultptr, optype, ptype, param;                               01662
    CASE parsemode OF                                                  01663
       = parsing:                                                      01664
          CASE optype OF                                               01665
             = 102 %-quickprint-%:                                     01666
                CASE ptype OF                                          01667
                   = 135 %- right -%:      uqpcolmax _
                   getpint(&param, &param+d2sel);                      01668
                   ENDCASE typeas($"invalid option specified");
                                                                       01669
             ENDCASE typeas($"invalid option specified");              01670
       ENDCASE;                                                        01671
    RETURN( &resultptr );                                              01672
    END.
                                                                       01673
(xuoprint)        % set print parameters %                             0150
```

```
    PROCEDURE (resultptr, parsemode, type, param);                  0151
    LOCAL tp, da, end;                                              0152
    LOCAL STRING temp[200];                                         0153
    REF resultptr, type, param, tp, da;                             0154
    CASE parsemode OF                                               0155
       = parsing:                                                   0156
          CASE type OF                                              0157
             = 135 %- right -%:    colmax _ getpint(&param,
             &param+d2sel);                                         0158
             = 136 %- left -%:     tpoffset _ getpint(&param,
             &param+d2sel);                                         0159
             = 186 %- bottom -%:   linmax _ getpint(&param,
             &param+d2sel);                                         0160
             = 187 %- page -%:     pgsize _ getpint(&param,
             &param+d2sel);                                         0161
             = 182 %- indenting -%:                                 0162
                BEGIN                                              01354
                indcnt _ getpint(&param, &param+d2sel);            01352
                end _ (&da _ $dpyarea) + dal*dacnt;                01355
                DO IF da.daexis THEN da.daind _ indcnt             01356
                UNTIL (&da _ &da + dal) >= end;                    01357
                END;                                               01353
             = 166 %- tab -%:                                       0163
                BEGIN                                               0164
                &tp _ &param + d2sel;                               0165
                *temp* _ param tp;                                  0166
                uotabs($temp);                                      0167
                end _ (&da _ $dpyarea) + dal*dacnt;                01358
                DO IF da.daexis THEN                               01359
                   BEGIN                                           01365
                   da.datab0 _ stdtab;                             01361
                   da.datab1 _ stdtab[1];                          01363
                   da.datab2 _ stdtab[2];                          01362
                   END                                             01364
                UNTIL (&da _ &da + dal) >= end;                    01360
                IF nlmode = typewriter THEN setabs($stdtab);       02329
                END;                                                0168
             ENDCASE typeas($"invalid option specified");           0169
       ENDCASE;                                                     0170
    RETURN( &resultptr );                                           0171
    END.
                                                                    0172
(xuoprompt)        % set prompt mode %                              0173
    PROCEDURE (resultptr, parsemode, type);                         0174
    REF resultptr, type;                                            0175
    CASE parsemode OF                                               0176
       = parsing:                                                   0177
          CASE type OF                                              0178
             = 188 %- off -%:      inprompt _ noprompts;            0179
             = 189 %- full -%:     inprompt _ fullprompts;          0180
             = 190 %- partial -%:  inprompt _ partprompts;          0181
             ENDCASE typeas($"invalid option specified");           0182
       ENDCASE;                                                     0183
    RETURN( &resultptr );                                           0184
    END.
                                                                    0185
```

```
(xuorecognition)  % set recognition mode %                          0186
   PROCEDURE (resultptr, parsemode, param, param2);                 0187
   REF resultptr, param, param2;                                    0188
   CASE parsemode OF                                                0189
      = parsing:                                                    0190
         CASE param OF                                              0191
            = 181 %- terse -%:                                      0192
               BEGIN                                                0193
               recogmode _ mexpert;                                 0194
               CASE param2 OF                                       0195
                  = 181 %- terse -%:        recog2mode _ mexpert;
                                                                    0196
                  = 191 %- anticipatory -%: recog2mode _
                  manticipatory;                                    0197
                  = 192 %- demand -%:       recog2mode _ mdemand;
                                                                    0198
                  = 193 %- fixed -%:        recog2mode _ mfixed;
                                                                    0199
                  ENDCASE typeas($"invalid option specified");
                                                                    0200
               END;                                                 0201
            = 191 %- anticipatory -%:                               0202
               recogmode _ recog2mode _ manticipatory;              0203
            = 192 %- demand -%:                                     0204
               recogmode _ recog2mode _ mdemand;                    0205
            = 193 %- fixed -%:                                      0206
               recogmode _ recog2mode _ mfixed;                     0207
            ENDCASE typeas($"invalid option specified");            0208
      ENDCASE;                                                      0209
   RETURN( &resultptr );                                            0210
   END.
                                                                    0211
(xuoreset)        % reset user options %                           01674
   PROCEDURE (resultptr, parsemode, param1, param2, param3);        01675
   REF resultptr, param1, param2, param3;                           01676
   CASE parsemode OF                                                01677
      = parsing:                                                    01678
         CASE param1 OF                                             01679
            = 95 %- all -%: uoreset();                              01680
            = 194 %- control -%:                                    01681
               BEGIN                                                01682
               uorescont();         % reset "cctbl" %              01683
               initch(nldevice);    % reset translate table %       01684
               initbtbl();          % and setup break table %       01685
               END;                                                 01686
            = 195 %- currentcontext -%:                             01687
               tslshchars _ defcurcontext;                          01688
            = 196 %- feedback -%:                                   01689
               CASE param2 OF                                       01690
                  = 99 %- mode -%:                                  01691
                     fbackmode _ defbmode;                          01692
                  = 85 %- length -%:                                01693
                     feedbk _ defdbk;                               01694
                  = 182 %- indenting -%:                            01695
                     fedind _ defdind;                              01696
                  ENDCASE typeas($"invalid option specified");
```

```
                                                                    01697
                = 197 %- herald -%:                                 01698
                   CASE param2 OF                                   01699
                      = 99 %- mode -%:                              01700
                         hrldmode _ defhmode;                       01701
                      = 85 %- length -%:                            01702
                         hrldsize _ defhsize;                       01703
                      ENDCASE typeas($"invalid option specified");
                                                                    01704
                = 18 %- name -%:                                    01705
                   dfnmdl _ dfnmdr _ 0;                             01706
                = 37 %- return -%:                                  01707
                   srrsize _ defsrrsize;                            01708
                = 36 %- filereturn -%:                              01709
                   frrsize _ deffrrsize;                            01710
                = 177 %- display -%:                                01711
                   CASE param2 OF                                   01712
                      = 135 %- right -%:                            01713
                         udpcolmax _ defdcolmax;                    01714
                      = 211 %- wraparound -%:                       02307
                         udpwrapcol _ defdwrapcol;                  02308
                      ENDCASE typeas($"invalid option specified");
                                                                    01715
                = 209 %- output -%:                                 01716
                   CASE param2 OF                                   01717
                      = 102 %- quickprint -%:                       01718
                        CASE param3 OF                              01719
                           = 135 %- right -%:                       01720
                              uqpcolmax _ defqcolmax;               01721
                           ENDCASE typeas($"invalid option
                           specified");                             01722
                      ENDCASE typeas($"invalid option specified");
                                                                    01723
                = 198 %- printoptions -%:                           01724
                   CASE param2 OF                                   01725
                      = 135 %- right -%:                            01726
                         colmax _ defcolmax;                        01727
                      = 136 %- left -%:                             01728
                         tpoffset _ defoffset;                      01729
                      = 186 %- bottom -%:                           01730
                         linmax _ deflinmax;                        01731
                      = 187 %- page -%:                             01732
                         pgsize _ defpgsize;                        01733
                      = 182 %- indenting -%:                        01734
                         indcnt _ defindcnt;                        01735
                      = 166 %- tab -%:                              01736
                         BEGIN                                      01737
                         stdtab _ deftb1;                           01738
                         stdtab[1] _ deftb2;                        01739
                         stdtab[2] _ deftb3;                        01740
                         END;                                       01741
                      ENDCASE typeas($"invalid option specified");
                                                                    01742
                = 199 %- prompt -%:                                 01743
                   inprompt _ defprompt;                            01744
                = 200 %- recognition -%:                            01745
```

```
                BEGIN                                                   01746
                recogmode _ defrecmode;                                 01747
                recog2mode _ defre2mode;                                01748
                END;                                                    01749
             = 210 %- space for tab -%:                                 01963
                <uoresspfortab>();                                      01968
             = 112 %- viewspecs -%:                                     01750
                BEGIN                                                   01751
                stdvsp _ defvs1;                                        01752
                stdvsp[1] _ defvs2;                                     01753
                novspec _ FALSE;                                        01754
                END;                                                    01755
             = 201 %- startup -%:                                       01756
                stupstr _ defstu;                                       01757
             = 113 %- external -%:                                      01758
                enlfstr _ defenlf;                                      01759
             = 184 %- entry -%:                                         01760
                *usys2* _ *dsys2*;                                      01761
             = 126 %- default -%:                                       01762
                uoresubs();                                             01763
             = 183 %- universal -%:                                     01764
                *usys1* _ *dsys1*;                                      01765
             ENDCASE typeas($"invalid option specified");               01766
          ENDCASE;                                                      01767
       RETURN( &resultptr );                                            01768
       END.
                                                                        01769
    (xuoringsize)     % set return ring size %                          0134
       PROCEDURE (resultptr, parsemode, type, param);                   0135
       REF resultptr, type, param;                                      0136
       CASE parsemode OF                                                0137
          = parsing:                                                    0138
             CASE type OF                                               0139
                = 37 %- return -%:                                      0140
                   srrsize _ MAX( 1,                                    0141
                      MIN(getpint(&param, &param+d2sel), srrmax));
                                                                        0142
                = 36 %- filereturn -%:                                  0143
                   frrsize _ MAX( 1,                                    0144
                      MIN(getpint(&param, &param+d2sel), frrmax));
                                                                        0145
             ENDCASE typeas($"invalid option specified");               0146
          ENDCASE;                                                      0147
       RETURN( &resultptr );                                            0148
       END.
                                                                        0149
    (xuoshow) % show user options %                                    01770
       PROCEDURE (resultptr, parsemode, param1, param2);                01771
       REF resultptr, param1, param2;                                   01772
       CASE parsemode OF                                                01773
          = parsing:                                                    01774
             BEGIN                                                      01775
             *lit* _ NULL;                                              01776
             CASE param1 OF                                             01777
                = 95 %- all -%: uoshow();                               01778
                = 194 %- control -%: IF param2 = 95 %+ all +% THEN
```

```
            uoshctrl()                                                  01779
               ELSE uoshcc(strdev(&param2), TRUE);                      01780
            = 195 %- currentcontext -%: uoshcurc();                     01781
            = 196 %- feedback -%: uoshfeed();                           01782
            = 197 %- herald -%: uoshrld();                              01783
            = 37 %- return -%: uoshjmp();                               01784
            = 202 %- leveladjust -%: uoshlev();                         01785
            = 18 %- name -%: uoshnmd();                                 01786
            = 177 %- display -%: uoshdisplay();                         01787
            = 209 %- output -%: uoshoutput();                           01788
            = 198 %- printoptions -%: uoshprint();                      01789
            = 199 %- prompt -%: uoshpmt();                              01790
            = 200 %- recognition -%: uoshreco();                        01791
            = 112 %- viewspecs -%: uoshview();                          01792
            = 210 %- space for tabs -%: uoshspftb();                    01793
            = 201 %- startup -%: uoshstup();                            02071
            = 113 %- external -%: uoshextn();                           01794
            = 184 %- entry -%: uoshesub();                              01795
            = 126 %- default -%: uoshisubs();                           01796
            = 183 %- universal -%: uoshssub();                          01797
            ENDCASE typeas($"invalid option specified");                01798
         fbctl(typecalit, $lit);                                        01799
         END;                                                           01800
      ENDCASE;                                                          01801
   RETURN( &resultptr );                                                01802
   END.
                                                                        01803
(xuospfortab)     % set space for tab flags %                           02235
   PROCEDURE (resultptr, parsemode, dent, sent, source);                02236
   REF resultptr, dent, sent, source;                                   02237
   CASE parsemode OF = parsing:                                         02238
      BEGIN                                                             02239
      spftab _ dent;                                                    02240
      IF nlmode # fulldisp THEN                                         02241
         BEGIN                                                          02242
         r1 _ 18M;                                                      02243
         r2 _ IF spftab THEN 12737B3 ELSE 10737B3;                      02244
         r3 _ 1B3;                                                      02245
         !JSYS sfcoc;                                                   02246
         END;                                                           02247
      IF (rtjtab _ sent) AND spftab AND source THEN                     02248
         BEGIN                                                          02249
         <getpstring>(&source,$lit);                                    02250
         % in case useroption page has a 0 for rjtchr %                 02254
            rjtchr.L _ 0;                                               02255
            rjtchr.M _ 5;                                               02256
         IF lit.L > 5 THEN                                              02251
            BEGIN                                                       02252
            lit.L _ 5;                                                  02253
            *rjtchr* _ *lit*;                                           02257
            *lit* _ "Only five termination characters allowed.
            You currently have:  ", *rjtchr*;                           02258
            err($lit)                                                   02259
            END                                                         02260
         ELSE *rjtchr* _ *lit*;                                         02261
         END;                                                           02262
```

```
      END;                                                          02263
      ENDCASE;                                                      02264
   RETURN( &resultptr );                                            02265
   END.
                                                                    02266
(xuostup) % set startup commands branch addressF %                   0866
   PROCEDURE ( resultptr, parsemode, param);                        0867
   LOCAL TEXT POINTER tp1;                                          0904
   LOCAL adstr[40];                                                 0898
   LOCAL STRING locstr[200];                                        0907
   REF resultptr, param ;                                           0868
   CASE parsemode OF                                                0869
      = parsing:                                                    0870
         BEGIN                                                      0880
         IF param.stfile THEN                                       0881
            BEGIN                                                   0882
            lnkprs( &param, $adstr);                                0899
            param _ adstr[ls];                                      0900
               param[1] _ adstr[ls+1];                              0901
            [&param+d2sel] _ adstr[le];                             0902
               [&param+d2sel+1] _ adstr[le+1];                      0903
            END;                                                    0891
         tp1 _ [&param+d2sel];                                      0905
            tp1[1] _ [&param+d2sel+1];                              0906
         %initialize string first time%                             0934
            IF NOT stupstr.M THEN stupstr _ defstu;                 0933
         *locstr* _ *stupstr*;                                      0908
         ON SIGNAL ELSE                                             0909
            BEGIN                                                   0910
            ON SIGNAL ELSE;                                         0932
            *stupstr* _ *locstr*;                                   0911
            *locstr* _ "address must be less than ",                0912
               STRING(stupstr.M), "characters long";                0914
            err( $locstr);                                          0915
            END;                                                    0916
         *stupstr* _ param tp1;                                     0917
         END;                                                       0897
      ENDCASE;                                                      0875
   RETURN( &resultptr);                                             0876
   END.                                                             0877
                                                                    0878
(xuoviewspecs)    % set standard viewspecs %                        0212
   PROCEDURE (resultptr, parsemode, param, vs);                     0213
   LOCAL char, temp;                                                0214
   REF resultptr, param, vs, temp;                                  0215
   CASE parsemode OF                                                0216
      = parsing:                                                    0217
         CASE param OF                                              0816
            = 126 %- default -%:                                    0817
               BEGIN                                                0218
               &temp _ &vs + 3;                                     0219
               CCPOS SF(*temp*);                                    0220
               CASE char _ READC OF                                 0221
                  = ENDCHR : NULL;                                  0222
                  ENDCASE                                           0223
                     BEGIN                                          0224
```

```
                        stdvsp _ setlt(char,stdvsp,stdvsp[1] :
                        stdvsp[1]);                                    0225
                        REPEAT CASE;                                   0226
                        END;                                           0227
                    END;                                               0228
                = 1 : novspec _ FALSE;                                 0818
                = 2 : novspec _ TRUE;                                  0819
                ENDCASE typeas($"invalid option specified")            0820
        ENDCASE;                                                       0229
    RETURN( &resultptr );                                              0230
    END.
                                                                       0231
% lower level support routines %                                       0327
  (uoreset) % reset all user options %                                01840
    PROCEDURE ;                                                       01841
    % this procedure resets all the user options to the system
    standards. It uses the data at (nls,const,defuseroptions) %
                                                                      01842
    % control characters %                                           01843
      uorescont();          % reset "ctbl" %                        01844
      initch(nldevice);     % reset translate tables %              01845
      initbtbl(); % and break table %                               01846
    % current context %                                             01847
      tslshchars _ defcurcontext;                                   01848
    % feedback %                                                    01849
      fbackmode _ defbmode;                                         01850
      feedbk _ defdbk;                                              01851
      fedind _ defdind;                                             01852
    % herald %                                                      01853
      hrldmode _ defhmode;                                          01854
      hrldsize _ defhsize;                                          01855
    %name delimiters%                                               01856
      dfnmdl _ dfnmdr _ 0;                                          01857
    % jump rings %                                                 01858
      srrsize _ defsrrsize;                                         01859
      frrsize _ deffrrsize;                                         01860
    % display %                                                     01861
      udpcolmax _ defdcolmax;                                       01862
      udpwrapcol _ defdwrapcol;                                     02306
    % output %                                                      01863
      % quickprint %                                               01864
        uqpcolmax _ defqcolmax;                                     01865
    % print options %                                               01866
      colmax _ defcolmax;                                           01867
      tpoffset _ defoffset;                                         01868
      linmax _ deflinmax;                                           01869
      pgsize _ defpgsize;                                           01870
      indcnt _ defindcnt;                                           01871
      stdtab _ deftb1;                                              01872
      stdtab[1] _ deftb2;                                           01873
      stdtab[2] _ deftb3;                                           01874
    % prompt %                                                      01875
      inprompt _ defprompt;                                         01876
    % recognition %                                                01877
      recogmode _ defrecmode;                                       01878
      recog2mode _ defre2mode;                                      01879
```

```
    % space for tabs %                                                  01961
      uoresspfortab();                                                  02274
    % viewspecs %                                                       01880
      stdvsp _ defvs1;                                                  01881
      stdvsp[1] _ defvs2;                                               01882
      novspec _ FALSE; %default is viewspec promting on%                01883
    % null string address for startup process commands %               01884
      stupstr _ defstu;                                                 01885
    % null string address for external names link file address %
                                                                        01886
      enlfstr _ defenlf;                                                01887
    % reset the automatically loaded subsystems and programs%          01888
      uoresub();                                                        01889
    RETURN;                                                             01890
    END.
                                                                        01891
(uoresub) %reset initially loaded subsystems and programs%              01309
    PROCEDURE ;                                                         01310
    LOCAL pusys,pdsys,numw,inc;                                         01311
    REF pusys,pdsys;                                                    01312
      usys1 _ defsubstr;                                                01346
      numw _ (usys1.M + 4)/5 + 1;                                       01313
      &pdsys _ $dsys1;                                                  01314
      FOR &pusys _ $usys1 UP numw UNTIL > $usys15 DO                    01315
        BEGIN                                                           01316
        % set up correct maximum length for each string and store
        default%                                                        01344
        pusys _ defsubstr;                                              01345
        *pusys* _ *pdsys*;                                              01317
        % L10 compiler drops out a zero word after initialized
        strings !%                                                      01348
        inc _ IF pdsys.L THEN 5 ELSE 4;                                 01347
        &pdsys _ &pdsys + (pdsys.M + inc)/5 + 1;                        01318
        END;                                                            01319
    RETURN(TRUE);                                                       01320
    END.                                                                01321
(uoresspfortab) PROC ; % resets space for tab status to default.
Currently OFF %                                                         02271
    spftab _ rtjtab _ FALSE;                                            02276
    RETURN;                                                             02322
    END.
                                                                        02277
(uoget)          % get the user profile page(s) %                        0366
    PROCEDURE ;                                                         0367
    LOCAL STRING filnms[100];                                           0368
     % this guy opens the user-options file for write and maps it
    into the users address space. It overlays the current private
    (read only) page %                                                  0369
    % get a jfn for file <logindir>PROFILE,ident;1 %                    0370
      *filnms* _ '<,  *userstr*, '>, "$USER-PROFILE$.", *initsr*,
      fvrdchar, "1", 0;                                                 0371
      IF NOT SKIP !gtjfn(1B6,$filnms+chbmty) THEN                       0372
        BEGIN                                                           0373
        typeas($" can not get your PROFILE??
         ** using system defaults **");                                 0374
        RETURN(FALSE);                                                  0375
```

```
        END;                                                    0376
      uojfn _ r1;                                               0377
    % open it 36 bits read, write and thawed %                  0378
      IF NOT SKIP !openf(uojfn,440000302000B) THEN              0379
        BEGIN                                                   0380
        % open it 36 bits read if can't for write %             0381
        IF NOT SKIP !openf(uojfn,4400002020008) THEN            0382
          BEGIN                                                 0383
          IF NOT SKIP !rljfn(uojfn := 0) THEN NULL;             0384
          typeas($" can not open your PROFILE file??
           ** using system defaults **");                       0385
          RETURN(FALSE);                                        0386
          END;                                                  0387
        % get a private copy and close file %                   0388
          r1.LH _ uojfn;          % profile file jfn %          0389
          r1.RH _ 0;              % page zero %                 0390
          r2.LH _ 400000B;        % this fork %                 0391
          r2.RH _ $userdata / 1000B;                            0392
          r3 _ cwacc .V racc;     % read/copy-on-write access %
                                                                0393
          !pmap();                                              0394
          userdata _ FALSE;       % make if private %           0395
          IF NOT SKIP !closf(uojfn) THEN NULL;                  0396
          uojfn _ 0;              % say open for read %         0397
          typeas( $"USER-PROFILE opened read only!!");          0803
          RETURN(TRUE); % say we got user profile %             0398
        END;                                                    0399
    % turn on archive "don't delete" bit in fdb %              02341
      chnfdb(uojfn, 178, 1B10, 1B10);                           02342
    % map page into users address space at "userdata" %        0400
      r1.LH _ uojfn;          % profile file jfn %              0401
      r1.RH _ 0;              % page zero %                     0402
      r2.LH _ 400000B;        % this fork %                     0403
      r2.RH _ $userdata / 1000B;                                0404
      r3 _ racc .V wacc;  %read and write access %              0405
      !pmap();                                                  0406
    % initialize if nessecary %                                0407
      IF NOT userdata THEN                                      0408
        BEGIN                                                   0409
        uoreset();  % initialize to system defaults %           0410
        userdata _ TRUE;  % say initialized %                   0411
        END;                                                    0412
    % initialize automatically loaded subsystems if nessecary %
                                                                01349
      IF usys1.M = 0 THEN uoresub();                            01350
      RETURN(TRUE);                                             0413
    END.                                                        0414
(uoincnam) PROC (param,uoifile) ; % given a ptr to a fileaddress
for the first parameter, update the string whose address is the
second parameter with a filename that will fit in the available
space %                                                         01609
    LOCAL adstr[40];                                            01610
    LOCAL STRING errstr[200];                                   01611
    LOCAL TEXT POINTER dp1,dp2, tp1,tp2,tp3;                    01612
    REF param, uoifile;                                         01613
    inkprs(&param,$adstr);                                      01614
```

```
    dp1 _ adstr[us]; dp1[1] _ adstr[us+1];                              01615
    dp2 _ adstr[ue]; dp2[1] _ adstr[ue+1];                              01616
    tp1 _ adstr[fs]; tp1[1] _ adstr[fs+1];                              01617
    tp2 _ adstr[fe]; tp2[1] _ adstr[fe+1];                              01618
    %get rid of extension and or version number%                        01619
    IF NOT FIND BETWEEN tp1 tp2 ( [ '. / ';] ^tp3 _tp3 ) THEN FIND
    tp2 ^tp3;                                                           01620
    IF dp1[1] < dp2[1] THEN *uoifile* _ +dp1 dp2, ',, +tp1 tp3 ELSE
    *uoifile* _ +tp1 tp3;                                               01621
    IF uoifile.L > usys1.M THEN                                         01622
       BEGIN                                                            01623
       *uoifile* _ +tp1 tp3;                                            01624
       IF uoifile.L > usys1.M THEN                                      01625
          BEGIN                                                         01626
          *errstr* _ "Program name must be less than ",
          STRING(usys1.M) , " characters long";                         01627
          err($errstr);                                                 01628
          END;                                                          01629
       *errstr* _ "More than ", STRING(usys1.M) , " characters long
       so filename used without directory.";                            01630
       dismes(1,$errstr);                                               01631
       END;                                                             01632
    RETURN;                                                             01633
    END.
                                                                        01634
(uoclose) % close the user options file %                                0415
    PROCEDURE ;                                                         0416
     % this guy maps out the user-options page(s) into the PROFILE
    file and closes the file %                                          0417
    % unmap "userdata" page %                                          0418
       r1 _ -1;                                                         0419
       r2.LH _ 400000B;                                                 0420
       r2.RH _ $userdata / 1000B;                                       0421
       r3 _ 0;                                                         02344
       !pmap();                                                         0422
    % close the file %                                                  0423
       IF NOT SKIP !closf(uojfn) THEN NULL;                             0424
    RETURN;                                                             0425
    END.                                                                0426
(uoaccess)        % sets the access to user options file %               0427
    PROCEDURE(pagno, access) ;                                          0428
    % this guy switches the access for the user option page(s) to
    thE passed access %                                                 0429
    r1.LH _ 400000B;       % this fork %                                0430
    r1.RH _ pagno;                                                      0431
    r2 _ access;                                                        0432
    !spacs();                                                           0433
    RETURN;                                                             0434
    END.                                                                0435
(uotblget)        % get a "free" entry in "cctbl" %                     0436
    PROCEDURE ;                                                         0437
    LOCAL i;                                                            0438
    FOR i _ 0 UP UNTIL = 100 DO IF cctbl[i] = 0 THEN RETURN(i);
                                                                        0439
    err($"only 100 non-standard ctrlchar definitions allowed")          0440
    END.                                                                0441
```

```
(uorescont)       % reset nonstandard ctrlchar table %          0442
   PROCEDURE ;                                                  0443
   LOCAL i, j, tblwd, ttydevs[10];                              0444
   % initialize tty devices array %                             0445
      ttydevs _ dev33;                                          0446
      ttydevs[1] _ dev35;                                       0447
      ttydevs[2] _ dev37;                                       0448
      ttydevs[3] _ devtiex;                                     0449
      ttydevs[4] _ nettty;                                      0450
      ttydevs[5] _ -1;    % end of table %                       0451
   % set nonstandard ctrlchar table = NULL %                    0452
      FOR i _ 0 UP UNTIL = 100 DO cctbl[i] _ 0 ;                0453
   % set typewriter specific characters %                       0454
      % command accept %                                       0455
         tblwd.cctype _ CA;                                     0456
         tblwd.ccchar _ EOL;                                    0457
         tblwd.ccecho _ nullch;                                 0458
         i _ 0;                                                 0459
         LOOP                                                   0460
            BEGIN                                               0461
            IF ttydevs[i] = -1 THEN EXIT LOOP;                  0462
            j _ uotblget();       % get a free entry %          0463
            tblwd.ccdevice _ ttydevs[i];                        0464
            cctbl[j] _ tblwd;                                   0465
            i _ i + 1;                                          0466
            END;                                                0467
      % repeat commented out leave as default <^B>              01003
         tblwd.cctype _ C.;                                     01004
         tblwd.ccchar _  $ascalt                                01005
         tblwd.ccecho _ 2B;                                     01006
         i _ 0;                                                 01007
         LOOP                                                   01008
            BEGIN                                               01009
            IF ttydevs[i] = -1 THEN EXIT LOOP;                  01010
            j _ uotblget();                                     01011
            tblwd.ccdevice _ ttydevs[i];                        01012
            cctbl[j] _ tblwd;                                   01013
            i _ i + 1;                                          01014
            END;%                                               01015
   RETURN;                                                      0481
   END.                                                         0482
(uotabs)          % parse user string and set tabs %           02130
   PROCEDURE                                                    02131
      (tabstr);   % pointer to tab string %                     02132
   LOCAL tp, i, form, cpos, char1, char2, char3, tabarray[132];
                                                                02133
   LOCAL TEXT POINTER tp1, tp2, tp3;                            02134
   LOCAL STRING temp[200], numstr[10];                          02135
   REF tabstr;                                                  02136
   % 1st determine if string is one of standard forms %         02137
      CCPOS SF(*tabstr*);                                       02138
      IF NOT FIND $CR (SP/ ', / D) THEN err($" invalid user tab
      string format"); % TNLS user needs to be able to place CR at
      beginning %                                               02139
      form _ 0;   % default to invalid form %                   02140
      CCPOS SF(*tabstr*);                                       02141
```

```
        IF FIND $(SP/TAB) ^tp1 $D $(SP/TAB) ^tp2 ', $(SP/TAB) ^tp3
        THEN                                                        02142
           BEGIN                                                    02143
           CCPOS tp1;                                               02144
           char1 _ READC;   % get relavent characters %             02145
           CCPOS tp2;                                               02146
           char2 _ READC;                                           02147
           CCPOS tp3;                                               02148
           char3 _ READC;                                           02149
           CASE char1 OF                                            02150
              = D:                                                  02151
                 CASE char2 OF                                      02152
                    = ',:                                           02153
                       CASE char3 OF                                02154
                          = D: form _ 2;                            02155
                          ENDCASE form _ 1;                         02156
                    ENDCASE form _ 1;                               02157
              ENDCASE form _ 1;                                     02158
           END                                                      02159
        ELSE                                                        02160
           IF (FIND SF(*tabstr*) $D) AND (NOT FIND [SP]) THEN form _
           2                                                        02343
           ELSE IF FIND SF(*tabstr*) $(LD/SP/TAB) THEN form _ 1;
                                                                    02161
     % now build tab array and go build tab words %                 02162
        CCPOS SF(*tabstr*);                                         02163
        tabarray _ 0;                                               02164
        cpos _ 1;                                                   02165
        CASE form OF                                                02166
           = 1: % FORM 1 - "  c      c    c   ...etc..." %        02167
              FOR i _ 0 UP UNTIL =10 DO % ten tabs for now %        02168
                 LOOP                                               02169
                    CASE READC OF                                   02170
                       = CR:  %tnls users need to be able to enter
                       CR %                                         02171
                          BEGIN                                     02172
                          cpos _ 1;                                 02173
                          REPEAT CASE;                              02174
                          END;                                      02175
                       = ENDCHR: EXIT LOOP 2;                       02176
                       = SP: cpos _ cpos + 1;                       02177
                       = TAB:                                       02178
                          BEGIN                                     02179
                          % fancy shit goes here %                  02180
                          END;                                      02181
                       ENDCASE                                      02182
                          BEGIN                                     02183
                          tabarray[i] _ cpos;                       02184
                          tabarray[i + 1] _ 0;                      02185
                          cpos _ cpos + 1;                          02186
                          EXIT LOOP;                                02187
                          END;                                      02188
           = 2: % FORM " - "colnum(i),colnum(j),colnum(k),....." %
                                                                    02189
              BEGIN                                                 02190
              tabarray[0] _ 0;                                      02191
```

```
            FOR i _ 0 UP UNTIL =10 DO                                   02192
               BEGIN                                                    02193
               IF NOT FIND $(SP/TAB) ^tp1 $D ^tp2 THEN EXIT;            02194
               *numstr* _ tp1 tp2;                                      02195
               tabarray[i] _ VALUE($numstr);                            02196
               tabarray[i+1] _ 0;                                       02197
               IF NOT FIND $(SP/TAB) ", ^tp1 THEN EXIT;                 02198
               END;                                                     02199
            END;                                                        02200
         ENDCASE err($" invalid user tab string format");               02201
      bildtabwrds($tabarray, $stdtab);                                  02202
      RETURN;                                                           02203
   END.                                                                 02204
(bildtabwrds)     % builds 3word tab table %                            0551
   PROCEDURE                                                            0552
      (array,     %address of array of tab values%                      0553
      tbl);                %address of table to set%                   0554
   LOCAL val, i;                                                        0555
   REF array, tbl;                                                      0556
   i_0;                                                                 0557
   dassnbit($stdtab, -1, 3); %clear all tabs%;                          0558
   WHILE val _ array[i := i+1] DO setbit($stdtab, val, 3);              0559
   RETURN;                                                              0560
   END.                                                                 0561
(getpstring) PROC (aptr, astring);  % given an address to a
pointer to a string, and the address of a string, puts the pointed
to string into the addressed string. %                                 01951
   REF aptr, astring;                                                   01952
   LOCAL TEXT POINTER ptr1, ptr2;                                       01953
   ptr1 _ aptr;  ptr1[1] _ aptr[1];                                     01954
   ptr2 _ aptr[d2sel];  ptr2[1] _ aptr[d2sel+1];                        01955
   *astring* _ ptr1 ptr2;                                               01956
   RETURN;                                                              01957
   END.
                                                                       01958
% "SHOW" support routines %                                             0562
(uolitap) PROCEDURE (string); %used by uoshow, etc, to cut down
space%                                                                 02345
   REF string;                                                         02346
   *lit* _ *lit*, *string*;                                            02347
   RETURN; END.                                                        02348
(uolchap) PROCEDURE (char); %used by uoshow, etc, to cut down
space%                                                                 02360
   *lit* _ *lit*, char;                                                02362
   RETURN; END.                                                        02363
(uolcrap)PROCEDURE(string); %append CRLF, string to lit%               02349
   REF string;                                                         02350
   *lit* _ *lit*, CR, LF, *string*;                                    02351
   RETURN; END.                                                        02352
(uolstrap)PROCEDURE(value);                                            02394
   *lit* _ *lit*, STRING(value);                                       02398
   RETURN; END.                                                        02396
(uoshow)  % show entire user options data %                            01804
   PROCEDURE ;                                                         01805
   *lit* _ NULL;                                                       01806
   uoshctrl();                                                         01807
```

```
    uolcrap($"Herald"); uoshrld();                                    01809
    uolcrap($"Current Context"); uoshcurc();                          01811
    uolcrap($"Feedback"); uoshfeed();                                 01813
    uolcrap($"Prompt"); uoshpmt();                                    01815
    uolitap($"  Recognition"); uoshreco();                            01817
    uolcrap($"Jump"); uoshjmp();                                      01819
    uolcrap($"Name Delimiters"); uoshnmd();                           01821
                                                                      01822
    fbctl(norstcalit, $lit);                                          01823
                                                                      01824
    *lit* _ NULL;                                                     01825
    uoshlev();                                                        01826
    uolcrap($"Print Options"); uoshprint();                           01828
    uolcrap($"Space for Tab"); uoshspftb();                           02073
    uolcrap($"Output"); uoshoutput();                                 01830
    uolitap($"  Display"); uoshdisplay();                             01832
    uolcrap($"Startup Commands Branch Address: "); uoshstup();
                                                                      01834
    uoshview();                                                       01835
    uoshextn();                                                       01836
    uoshisubs(); %initial subsystems%                                 01837
    RETURN;                                                           01838
    END.                                                              01839
(uoshctrl)        % displays U-OP ctrlchar definitions %             0585
    PROCEDURE ;                                                       0586
    LOCAL maxdev, i, j, char, cchar, foundf;                          0587
    uoshsc();                                                         0994
    FOR i _ 0 UP UNTIL = 10B DO uoshcc(i, FALSE);                     0589
    RETURN;                                                           0590
    END.                                                              0591
(uoshsc)  % display standard ctrlchar definitions %                   0992
    PROCEDURE;                                                        0993
    uolitap($"Control Characters: Standard Definitions
    (non-alterable)");                                                02353
    uolcrap($"  CA:<^D>, CD:<^X>, RPT:<^B>, INSERT:<^E>, BC:<^A>,
    BW:<^W>,");                                                       02354
    uolcrap($"  BS:<^K>, LITESC:<^V>, IGNCRE:<^$>, SC:<^$>,
    SW:<^$>, TAB:<^I>");                                              02355
    uolcrap($"Control Characters: User Definitions (alterable)");
                                                                      02356
    RETURN;                                                           0995
    END.                                                              0996
(uoshcc)  % display control chars for specific terminal %             0592
    PROCEDURE(device, initmode);                                      0593
    LOCAL foundf, j, char, cchar, cecho;                              0594
    IF initmode THEN uoshsc();                                        0997
    foundf _ FALSE;                                                   0595
    FOR j _ 0 UP UNTIL = 100 DO                                       0596
       BEGIN                                                          0597
       IF cctbl[j] # 0 AND cctbl[j].ccdevice = device THEN            0598
          BEGIN                                                       0599
          char _ cctbl[j].ccchar;                                     0600
          cchar _ cctbl[j].cctype;                                    0601
          cecho _ cctbl[j].ccecho;                                    0602
          IF NOT foundf THEN                                          0603
             BEGIN                                                    0604
```

```
            uolcrap($"  "); uolitap(devstr(device));                    02357
            foundf _ TRUE;                                               0606
            END;                                                         0607
         uolchap(SP); uolitap(chrstr(cchar)); uolitap($":[");
                                                                         02359
         CASE char OF                                                    0609
            IN [SP, 'z] :                                                0610
               BEGIN uolchap(char); uolchap(',); END;                    02364
            ENDCASE                                                      0611
               BEGIN uolitap(npstrad(char)); uolchap(',); END;
                                                                         02365
         CASE char _ cecho OF                                            0612
            IN [SP, 'z] :                                                0613
               BEGIN uolchap(char); uolchap(']); END;                    02366
            ENDCASE                                                      0614
               BEGIN uolitap(npstrad(char)); uolchap(']); END;
                                                                         02367
         END;                                                            0615
      END;                                                               0616
   RETURN;                                                               0617
   END.                                                                  0618
(uoshcurc)        % displays U-OP current context length %               0619
   PROCEDURE ;                                                           0620
   uolitap($" length: "); uolstrap(tslshchars);                          02399
   RETURN;                                                               0622
   END.                                                                  0623
(uoshdisplay) % displays U-OP display parameters %                       02298
   PROCEDURE ;                                                           02299
   uolitap($" Right margin max: ");                                      02368
   uolstrap(udpcolmax);                                                  02300
   IF udpcolmax=0 THEN uolitap($" (zero means maximum)");                02301
   uolcrap($" Wraparound margin: ");                                     02369
   uolstrap(udpwrapcol);                                                 02400
   IF udpwrapcol=0 THEN uolitap($" (zero means no wraparound)");
                                                                         02303
   RETURN;                                                               02304
   END.                                                                  02305
(uoshesub)        % displays entry subsystem %                          01322
   PROCEDURE;                                                            01323
   uolcrap($"Entry subsystem: ");                                        02370
   IF enlfstr.L THEN uolitap($usys2)                                     01325
   ELSE uolitap($"Not Specified");                                       01326
   RETURN;                                                               01327
   END.                                                                  01328
(uoshssub)        % displays supervisor subsystem %                     01329
   PROCEDURE;                                                            01330
   uolcrap($"Universal Subsystem: ");                                    02371
   IF enlfstr.L THEN uolitap($usys1)                                     01332
   ELSE uolitap($"Not Specified");                                       01333
   RETURN;                                                               01334
   END.                                                                  01335
(uoshisubs)       % displays initial subsystems/character %             01336
   PROCEDURE;                                                            01337
   LOCAL i;                                                              01338
   uolcrap($"Universal Subsystem: "); uolitap($usys1);                   02372
   uolcrap($"Entry subsytem: "); uolitap($usys2);                        02373
```

```
    uolcrap($"Other Subsystems and User Programs: ");                    02374
    FOR i _ $usys3 UP ((usys1.M + 4)/5 + 1) UNTIL > $usys15 DO
                                                                         01340
      IF [i].L THEN BEGIN uolchap(' '); uolitap(i); END;                 01341
    RETURN;                                                              01342
    END.                                                                 01343
(uoshfeed)        % displays U-OP feedback paramters %                    0624
    PROCEDURE ;                                                          0625
    CASE fbackmode OF                                                    0626
      = verbsmode:                                                       0627
        BEGIN                                                            02377
        uolitap($" mode: VERBOSE, length: ");                            02379
        uolstrap(feedbk);                                                02401
        uolitap($" indenting: ");                                        02382
        uolstrap(fedind);                                                02402
        END;                                                             02378
      = tersemode:                                                       02383
        BEGIN                                                            02384
        uolitap($" mode: TERSE, length: ");                              02385
        uolstrap(feedbk);                                                02386
        uolitap($" indenting: ");                                        02387
        uolstrap(fedind);                                                02388
        END;                                                             02389
      ENDCASE err($"invalid feedback mode");                              0631
    RETURN;                                                              0632
    END.                                                                 0633
(uoshrld) % displays U-OP herald parameters %                            0634
    PROCEDURE ;                                                          0635
    CASE hrldmode OF                                                     0636
      = onechar:                                                         0637
        uolitap($" mode: TERSE");                                        02390
      = multchar:                                                        0639
        BEGIN                                                            02391
        uolitap($" mode: VERBOSE length: ");                             02393
        uolstrap(hrldsize);                                              02403
        END;                                                             02392
      ENDCASE err($"invalid herald mode");                               0641
    RETURN;                                                              0642
    END.                                                                 0643
(uoshjmp) % displays U-OP jump ring paramters %                          0644
    PROCEDURE ;                                                          0645
    uolitap($" return: ");                                           02405
    uolstrap(srrsize);                                               02404
    uolitap($" filereturn: ");                                       02406
    uolstrap(frrsize);                                               02407
    RETURN;                                                              0647
    END.                                                                 0648
(uoshlev) % displays U-OP level adjust prompting status %                0838
    PROCEDURE ;                                                          0839
    uolitap($"level adjust prompting: ");                            02408
    IF nolevadj THEN uolitap($" OFF")                                    0843
    ELSE uolitap($" ON");                                                0844
    RETURN;                                                              0841
    END.                                                                 0842
(uoshnmd) % displays U-OP name delimiter defaults %                      01393
    PROCEDURE ;                                                      01394
```

```
    IF dfnmdl = 0 THEN uolitap($" NULL ")                                01395
    ELSE                                                                 01396
       BEGIN                                                             02409
       uolchap(SP); uolchap(dfnmdl); uolchap(SP);                        02411
       END;                                                              02410
    IF dfnmdr = 0 THEN uolitap($" NULL ")                                01397
    ELSE                                                                 01398
       BEGIN                                                             02412
       uolchap(SP); uolchap(dfnmdr); uolchap(SP);                        02413
       END;                                                              02414
    RETURN;                                                              01399
    END.                                                                 01400
(uoshoutput) % displays U-OP output parameters %                         01635
    PROCEDURE ;                                                          01636
    LOCAL val;                                                           01637
    val _ IF uqpcolmax=0 THEN defqcolmax ELSE uqpcolmax;                 01638
    % so far, only quickprint right margin is in %                      01639
    uolcrap($" Quickprint: Right margin: "); uolstrap(val);              02415
    RETURN;                                                              01641
    END.                                                                 01642
(uoshprint)       % displays U-OP print option parameters %              0649
    PROCEDURE ;                                                          0650
    LOCAL tabno, sepchar;                                                0651
    uolcrap($"  Margins  left: "); uolstrap(tpoffset);                   02416
    uolitap($" right: "); uolstrap(colmax);                              02417
    uolitap($" bottom: "); uolstrap(linmax);                             02418
    uolcrap($"  page size: "); uolstrap(pgsize);                         02419
    uolitap($"  indenting per level: "); uolstrap(indcnt);               02420
    uolcrap($"  tabstops: ");                                            02421
    tabno _ 0;                                                           0654
    sepchar _ SP;                                                        0655
    WHILE tabno _ nxtbit($stdtabs, tabno, 3) DO                          0656
       BEGIN                                                             0657
       uolchap(sepchar); uolstrap(tabno);                                02422
       sepchar _ ",;                                                     0659
       END;                                                              0660
    RETURN;                                                              0661
    END.                                                                 0662
(uoshspfortab)    % displays U-OP space for tab parameters %             02074
    PROCEDURE ;                                                          02075
    IF spftab THEN                                                       02078
       BEGIN                                                             02091
       uolitap($"   ON");                                                02423
       IF rtjtab THEN                                                    02092
          BEGIN                                                          02095
          uolitap($"   Automatic backspacing ON");                       02424
          IF rjtchr.L THEN                                               02097
             BEGIN                                                       02427
             uolcrap($"        Termination characters besides CR
             and TAB:  ");                                               02425
             uolitap($rjtchr);                                           02426
             END                                                         02428
          ELSE uolitap($"        No termination characters other
          than CR and TAB.");                                            02103
          END                                                            02100
       ELSE uolitap($"   Automatic backspacing OFF.");                   02104
```

```
      END                                                          02093
   ELSE uolcrap($"   OFF");                                       02109
   RETURN;                                                        02086
   END.
                                                                  02087
(uoshpmt) % displays U-OP prompt parameters %                      0663
   PROCEDURE ;                                                     0664
   CASE inprompt OF                                                0665
      = noprompts: uolitap($" mode: OFF");                         0666
      = partprompts: uolitap($" mode: PARTIAL");                   02429
      = fullprompts: uolitap($" mode: FULL");                      02430
      ENDCASE err($"invalid prompt mode");                         0669
   RETURN;                                                         0670
   END.                                                            0671
(uoshreco)        % displays U-OP recognition parameters %         0672
   PROCEDURE ;                                                     0673
   CASE recogmode OF                                               0674
      = mexpert:                                                   0675
        BEGIN                                                      0676
        uolitap($" mode: TERSE ");                                 02431
        CASE recog2mode OF                                         0678
           = mexpert:                                              0679
              uolitap($" secondary mode: TERSE");                  02432
           = mfixed:                                               0681
              uolitap($" secondary mode: FIXED");                  02433
           = mdemand:                                              0683
              uolitap($" secondary mode: DEMAND");                 02434
           = manticipitory:                                        0685
              uolitap($" secondary mode: ANTICIPATORY");           02435
           ENDCASE err($"invalid secondary recognition mode");
                                                                   0687
        END;                                                       0688
      = mfixed:                                                    0689
        uolitap($" mode: FIXED ");                                 02436
      = mdemand:                                                   0691
        uolitap($" mode: DEMAND ");                                02437
      = manticipitory:                                             0693
        uolitap($" mode: ANTICIPATORY ");                          02438
      ENDCASE err($"invalid recognition mode");                    0695
   RETURN;                                                         0696
   END.                                                            0697
(uoshview)        % displays U-OP viewspecs %                      0698
   PROCEDURE ;                                                     0699
   uolcrap($"Viewspecs: ");                                        02439
   curvsp($stdvsp, $lit); % go convert to "human" form %           0701
   uolitap($"  prompting: ");                                      02440
   IF novspec THEN uolitap($" OFF")                                0846
   ELSE uolitap($" ON");                                           0847
   RETURN;                                                         0702
   END.                                                            0703
(uoshstup)        % displays U-OP startup commands branch address
%                                                                  0853
   PROCEDURE ;                                                     0854
   IF stupstr.L THEN uolitap($stupstr)                             0855
   ELSE uolitap($"Not Specified");                                 0862
   RETURN;                                                         0860
```

```
      END.                                                          0861
   (uoshextn)        % displays external names link file address %
                                                                    0936
      PROCEDURE;                                                    0937
      uolcrap($"External Names Link File Address: ");              02441
      IF enlfstr.L THEN uolitap($enlfstr)                           0939
      ELSE uolitap($"Not Specified");                               0940
      RETURN;                                                       0941
      END.                                                          0942
%.....string to internal code conversion....%                       0704
   (strchr)                                                         0705
      % converts string representation for character to internal code
      %                                                             0706
      PROCEDURE (cc);                                               0707
      LOCAL cchar;                                                  0708
      REF cc;                                                       0709
         CASE cc OF                                                 0710
            = 156 %- ca -%: cchar _ CA;                             0711
            = 157 %- cd -%: cchar _ CD;                             0712
            = 158 %- rpt -%: cchar _ C.;                            0713
            = 143 %- insert -%: cchar _ inschar;                    0714
            = 159 %- bc -%: cchar _ BC;                             0715
            = 160 %- bw -%: cchar _ BW;                             0716
            = 161 %- bs -%: cchar _ $ascbst;                        0717
            = 162 %- litesc -%: cchar _ 'V - 100B;                  0718
            = 163 %- ignore -%: cchar _ nullch;                     0719
            = 164 %- sc -%: cchar _ 0;                              0720
            = 165 %- sw -%: cchar _ 0;                              0721
            = 166 %- tab -%: cchar _ TAB;                           0722
            ENDCASE err($"undefined control character specified");
                                                                    0723
         RETURN(cchar);                                             0724
      END.                                                          0725
   (chrstr)                                                        02278
      % converts internal code for character to string representation
      %                                                            02279
      PROCEDURE (cc);                                              02280
      LOCAL chstrad;                                               02281
      CASE cc OF                                                   02282
         = CA: chstrad _ $"CA" %+ ca +% ;                          02283
         = CD : chstrad _ $"CD" %+ cd +% ;                         02284
         = C. : chstrad _ $"RPT" %+ rpt +% ;                       02285
         = inschar : chstrad _ $"INSERT" %+ insert +% ;            02286
         = BC : chstrad _ $"BC" %+ bc +% ;                         02287
         = BW : chstrad _ $"BW" %+ bw +% ;                         02288
         = $ascbst : chstrad _ $"BS" %+ bs +% ;                    02289
         = 'V - 100B : chstrad _ $"LITESC" %+ litesc +% ;          02290
         = nullch : chstrad _ $"IGNORE" %+ ignore +% ;             02291
         = 0 : chstrad _ $"SC" %+ sc +% ;                          02292
         = 0 : chstrad _  $"SW" %+ sw +% ;                         02293
         = TAB : chstrad _ $"TAB" %+ tab +% ;                      02294
         ENDCASE err($"undefined control character specified");    02295
      RETURN(chstrad);                                             02296
      END.                                                         02297
   (strdev)                                                         0746
      % converts string representation for device to internal code %
```

```
                                                                    0747
        % returns internal device code and nls mode %               0748
        PROCEDURE (dev);                                            0749
        LOCAL devcode, modetype;                                    0750
        REF dev;                                                    0751
        modetype _ typewriter; % default to tty %                   0752
        CASE dev OF                                                 0753
           = 205 %- tasker -% ,
           = 206 %- lineprocessor -% ,
           = 167 %- imlac -%:                                       0754
              BEGIN                                                 0757
              !gjinf();                                             0758
              !gttyp(r4 .V 4B5);                                    0759
              devcode _ r2;                                         0760
              CASE devcode OF                                       0761
                 = devlproc, = devsri, = imlac0, = imlac1 : NULL;   0762
                 ENDCASE err($"not a display terminal");            0763
              modetype _ fulldisplay;                               0764
              END;                                                  0765
           = 168 %- ti -%: devcode _ devtiex;                       0766
           = 169 %- nvt -%: devcode _ nettty;                       0767
           = 170 %- execuport -%: devcode _ devtiex;                0768
           % kludges for non-1st char numeric symbols %             0769
           = 33: devcode _ dev33;                                   0770
           = 35: devcode _ dev35;                                   0771
           = 37: devcode _ dev37;                                   0772
           ENDCASE err($"undefined device specified");              0773
        RETURN(devcode, modetype);                                  0774
        END.                                                        0775
     (devstr)                                                       0776
        % converts internal code for device to string representation %
                                                                    0777
        % returns address of string %                              0778
        PROCEDURE (dev);                                            0779
        LOCAL stradr;                                               0780
        CASE dev OF                                                 0781
           = devsri: stradr _ 205 %+ tasker +%;                     0782
           = devlproc: stradr _ 206 %+ lineprocessor +%;            0783
           = imlac0: stradr _ 167 %+ imlac +%;                      0784
           = imlac1: stradr _ 167 %+ imlac +%;                      0785
           = devtiex: stradr _ $"TI/EXECUPORT";                     0786
           = nettty: stradr _ 169 %+ nvt +%;                        0787
           % kludges for non-1st char numeric symbols %             0788
           = dev33: stradr _ $"33ASR";                              0789
           = dev35: stradr _ $"35ASR";                              0790
           = dev37: stradr _ $"37ASR";                              0791
           ENDCASE err($"undefined device specified");              0792
        RETURN(stradr);                                             0793
        END.                                                        0794
FINISH of psuserop                                                  0795
```

RECFIL

```
< NLS, RECFIL.NLS.3, >, 29-Mar-78 13:55 JDH ;;;; ["modeflg"];
FILE recfil  % L10 <rel-NLS>RECFIL %% (110,) (rel-nls,RECFIL.rel,) %  02
%Declarations%                                                        0782
   REGISTER r1 = 1, r2 = 2;                                            03
   DECLARE rectfg = 1, pc = 1, detflg = 0;                             04
   REF tda;                                                            05
   SET gjinf = 13B, deldf = 67B, gpjfn = 206B, spjfn = 207B;           06
(cxdetach) PROCEDURE ( rhosti, infile, rhosto, outfile );            01132
   LOCAL injfn, outjfn, savraw;                                      01133
   LOCAL STRING tempstr[100];                                        01134
   REF infile, outfile, rawchr;                                      01135
   CASE rhosti OF                                                    01136
      = lhostn: NULL;                                                01137
      ENDCASE err( $"remote file manipulations not implemented yet" );
                                                                     01138
   CASE rhosto OF                                                    01139
      = lhostn: NULL;                                                01140
      ENDCASE err($"Remote File Manipulations Not Implemented Yet");
                                                                     01141
   injfn _ outjfn _ 0;                                               01142
   *tempstr* _ "NIL:";                                               01143
   IF NOT &infile THEN &infile _ $tempstr;                           01144
   IF NOT &outfile THEN &outfile _ $tempstr;                         01145
   ON SIGNAL ELSE                                                    01146
      BEGIN                                                          01147
      IF injfn THEN                                                  01148
         BEGIN                                                       01149
         IF NOT SKIP !closf( injfn ) THEN NULL;                      01150
         IF NOT SKIP !rljfn( injfn := 0 ) THEN NULL;                 01151
         END;                                                        01152
      IF outjfn THEN                                                 01153
         BEGIN                                                       01154
         IF NOT SKIP !closf( outjfn ) THEN NULL;                     01155
         IF NOT SKIP !rljfn( outjfn := 0 ) THEN NULL;                01156
         END;                                                        01157
      END;                                                           01158
   IF NOT (injfn _ sgtjfn( gtjoif, &infile, $lit)) THEN              01159
      err( $"Can't GTJFN for input file" );                          01160
   IF NOT SKIP !openf( injfn, 7B10+2B5) THEN                         01161
      err( $"Can't OPEN input file");                                01162
   IF NOT (outjfn _ sgtjfn( gtjoof, &infile, $lit)) THEN             01163
      err( $"Can't GTJFN for output file" );                         01164
   IF NOT SKIP !openf( outjfn, 7B10+1B5) THEN                        01165
      err( $"Can't OPEN output file");                               01166
   IF nlmode = fulldisplay THEN                                      01167
      BEGIN                                                          01173
      savraw _ &rawchr;                                              01174
      xsimdev(lda(), devtiex, typewriter); % simulate a TI %         01168
      IF (&rawchr # savraw) AND (savraw = $auxchr) THEN              01175
         auxsav _ &rawchr := $auxchr;                                01177
      END;                                                           01178
   !spjfn( 400000B, (injfn * 1B6) + outjfn );                        01169
   !dtach();                                                         01170
   RETURN;                                                           01171
   END.
                                                                     01172
```

```
(nisutility) PROC(modeflg);                                          0862
   %Do utiity type of things for NLS%                                0863
   %---------------%                                                0864
   %Mode flg = 4 for run detatched, 5 for run attatched with separate
   primary output file, and 6 for run attatched with tty pimary output%
                                                                     0865
   LOCAL string, stid2, stid1, stid, stiddone, action, rtime, doany, fl,
   pmyjfn, pcap, da, savrub, savmrk, rubflg, newfile;                0866
   LOCAL TEXT POINTER tp1, tp2, tp3, tp4;                            0867
   LOCAL STRING wrkstr[100], filnms[50];                             0868
   REF string, fl, da;                                               0869
   tskerrcnt _ 0;                                                   01124
   stid _ stid1 _ 0;                                                 0870
   &da _ 0;                                                          0871
   ON SIGNAL                                                         0872
      =-5: %We Got here on a badfile, and who knows how%             0873
         IF stid.stfile = bfilno THEN                                0874
            BEGIN                                                    0875
            close(stid.stfile  := 0);                                0876
            typeas($"Task List File Bad");                           0877
            RETURN;                                                  0878
            END                                                      0879
         ELSE                                                        0880
            err($"Bad File");                                        0881
      ELSE                                                           0882
         BEGIN                                                      01125
         typeas(sysmsg);                                             0884
         typeas($"Fatal Error");                                     0885
         RETURN;                                                     0886
         END;                                                       01126
   %connect to NLS directory%                                        0888
      r1 _ 1;                                                        0889
      r2 _ chbptr(0) + $"NLS";                                       0890
      !JSYS stdir;                                                   0891
         GOTO conerr;                                                0892
         GOTO conerr;                                                0893
      r1.LH _ 0;                                                     0894
      r2 _ chbptr(0) + $"NLS";                                       0895
      IF NOT SKIP !JSYS cndir THEN                                   0896
         BEGIN                                                       0897
         (conerr):                                                   0898
         crlf();                                                     0899
         BUMP tskerrcnt;                                            01122
         typeas($"Directory Connect Fail");                          0900
         RETURN;                                                     0901
         END;                                                        0902
   %high queue this process if being run by a WHEEL or OPERATOR%     0903
      IF (pcap _ enablw()) NOT= -1 THEN                              0904
         BEGIN                                                       0905
         r1 _ 4B5;                                                   0906
         r2 _ 202B;                                                  0907
         !JSYS 243B;                                                 0908
         disablw(pcap);                                              0909
         END                                                         0910
      ELSE IF tenex < 13200 THEN typeas($"
      Unable to high queue -- not operator or wheel
```

```
   ");                                                                  0911
%Get tasks file %                                                       0912
   *filnms* _ "<NLS>TASKS.NLS";                                         0913
   stid _ orgstid;                                                      0914
   IF (stid.stfile _ open(0, $filnms : newfile)) = 0 THEN               0915
      BEGIN                                                             0916
      BUMP tskerrcnt;                                                  01123
      err($"Task File open fail");                                      0917
      END;                                                              0918
%set stid for "TODO" branch%                                            0919
   CCPOS SF(stid);                                                      0920
   FIND ^tp1;                                                           0921
   *wrkstr* _ "TODO";                                                   0922
   lookup($tp1, $wrkstr, nametyp);                                      0923
   IF tp1 = endfil THEN                                                 0924
      BEGIN                                                             0925
      typeas($"No ToDo Branch in Tasks File");                          0926
      RETURN;                                                           0927
      END;                                                              0928
%set stid for "DONE" branch (create if need to)%                        0929
   stid1 _ getsub(stid _ tp1);                                          0930
   IF stid1 = stid THEN                                                01127
      BEGIN                                                            01129
      dismes( 2, $"***  TASKS COMPLETED  ***");                        01130
      RETURN;                                                          01128
      END;                                                             01131
   *wrkstr* _ "DONE";                                                   0931
   lookup($tp1, $wrkstr, nametyp);                                      0932
   IF tp1 = endfil THEN                                                 0933
      BEGIN                                                             0934
      &string _ $"(DONE) Tasks Which Are Done";                         0935
      FIND SF(*string*) ^tp1 SE(*string*) ^tp2;                         0936
      stiddone _ cinssta(stid, levsuc, $tp1, $tp2);                     0937
      END                                                               0938
   ELSE stiddone _ tp1;                                                 0939
%setup rubout mechanism for local use%                                  0940
   rubflg _ FALSE;   % clear rubout flags %                             0947
   rubabt _ FALSE;                                                      0946
   IF modeflg # 4 THEN                                                  0941
      BEGIN                                                             0942
      savrub _ rubabt;                                                  0943
      savmrk _ rubmrk;        % save rubout flag address %             0944
      rubmrk _ $rubflg;       % set my own %                           0945
      END;                                                              0948
%Set up Primary Output File%                                            0949
   IF modeflg # 6 THEN                                                  0950
      pmyjfn _ setupop(0, $"U-OUT.TXT", modeflg, -1)                    0951
   ELSE pmyjfn _ 101B;        %use primary jfn%                         0952
%set to handle signals from compiler%                                   0953
   (utstat):                                                            0954
   ON SIGNAL ELSE                                                       0955
      BEGIN                                                             0956
      IF &da THEN delda(&da);                                           0957
      *datesr* _ NULL;                                                  0958
      stid2 _ cmovsta(stiddone,levdown, stid1 := getsuc(stid1),
      FALSE, 0);                                                        0959
```

```
      dpset(dspstrc, stid2, stid1, endfil);                          0960
      getdat($datesr); % time and date of completion %               0961
      *wrkstr* _ *datesr*, "   ", *[sysmsg]*;                        0962
      FIND SF(*wrkstr*) ^tp1 SE(*wrkstr*) ^tp2;                      0963
      stid2 _ cinsst(stid2, levdown, $tp1, $tp2);                    0964
      dpset(dspstrc,stid2, endfil, endfil);                          0965
      %Type out message if not detatched%                            0966
         IF modeflg # 4 THEN                                         0967
            BEGIN                                                    0968
            IF nlmode # fulldisplay THEN                             0969
               BEGIN                                                 0970
               *lit* _                                               0971
                  EOL,                                               0972
                  "**********************",                         0973
                  EOL,                                               0974
                  "**********************",                         0975
                  EOL,                                               0976
                  "    ", *[sysmsg]*, EOL,                           0977
                  "**********************",                         0978
                  EOL,                                               0979
                  "**********************", EOL, 0;                  0980
               !sout(pmyjfn, chbmty+$lit, 0);                        0981
               END                                                   0982
            ELSE % recreate display if tasks is loaded %             0983
               BEGIN                                                 0984
               IF NOT newfile THEN                                   0985
                  BEGIN                                              0986
                  recred();                                          0987
                  END                                                0988
               ELSE                                                  0989
                  BEGIN                                              0990
                  *lit* _ "** ", *[sysmsg]*, " **", 0;               0991
                  !sout(pmyjfn, chbmty+$lit, 0);                     0992
                  END;                                               0993
               END;                                                  0994
            END;                                                     0995
      GOTO utstat;                                                   0996
      END;                                                           0997
%get and setup a da for compilers to use%                            0998
   &da _ newda(); %get da for compile,print,or copy%                 0999
   intdafl(&da);                                                    01000
   da.davspec _ defvs1;                                             01001
   da.davspc2 _ defvs2;                                             01002
%finally process "TODO" branch performing functions%                01003
   LOOP                                                             01004
      BEGIN                                                         01005
      IF stid1 = stid OR rubflg THEN EXIT;                          01006
      %Now parse command statement%                                 01007
         CCPOS SF(stid1);                                           01008
         %Now, find what we are supposed to do%                     01009
            action _ IF FIND ["Compile"] THEN 1                     01010
               ELSE IF FIND ["Print"] THEN 2                        01011
               ELSE IF FIND ["Copy"] THEN 3                         01012
               ELSE 0;                                              01013
            IF action = 0 THEN err($"Undefined Command");           01014
         %Now get file name%                                        01015
```

```
            IF NOT FIND $NP ^tp1 1$PT ^tp2 THEN                          01016
               err($"Illegal Command Statement");                        01017
            *filnms* _ tp1 tp2;                                          01018
         %Set tp1 and tp2 to command%                                    01019
            FIND SF(stid1) ^tp1 SE(stid1) ^tp2;                          01020
      %Now execute%                                                      01021
         CASE action OF                                                  01022
            =1: %compile%                                                01023
               BEGIN                                                     01024
               IF modeflg # 4 THEN                                       01025
                  BEGIN                                                  01026
                  %Type out message to controlling tty%                  01027
                  *lit* _ EOL, EOL, "Compiling ", *filnms*, 0;           01028
                  !sout(pmyjfn, chbmty+$lit, 0);                         01029
                  END;                                                   01030
               runcmp($filnms, &da);                                     01031
               END;                                                      01032
            =2: %process print requests%                                 01033
               BEGIN                                                     01034
               IF modeflg # 4 THEN                                       01035
                  BEGIN                                                  01036
                  %Type out message to controlling tty%                  01037
                  *lit* _ EOL, EOL, "Printing ", *filnms*, 0;            01038
                  !sout(pmyjfn, chbmty+$lit, 0);                         01039
                  END;                                                   01040
               outptr($filnms, &da, 1, TRUE);                            01041
               END;                                                      01042
            =3: %process copy requests%                                  01043
               BEGIN                                                     01044
               IF modeflg # 4 THEN                                       01045
                  BEGIN %Type out message to controlling tty%            01046
                  *lit* _ 15B, 12B, "Copying ", *filnms*, 0;             01047
                  !sout(pmyjfn, chbmty+$lit, 0);                         01048
                  END;                                                   01049
               copsrc($filnms);                                          01050
               END;                                                      01051
            ENDCASE err($"Illegal Command");                             01052
         %Now fix up command statement and move it to done list%
                                                                         01053
            *datesr* _ NULL;                                             01054
            stid2 _ cmovsta(stiddone,levdown, stid1 := getsuc(stid1),
            FALSE, 0);                                                   01055
            dpset(dspstrc, stid2, stid1, endfil);                        01056
            getdat($datesr);  %date/time of complettion%                 01057
            *wrkstr* _ "Completed at " , *datesr*;                       01058
            FIND SF(*wrkstr*) ^tp1 SE(*wrkstr*) ^tp2;                    01059
            stid2 _ cinsst(stid2, levdown, $tp1, $tp2);                  01060
            dpset(dspstrc,stid2, endfil, endfil);                        01061
         % recreate display if tasks loaded and display %               01062
            IF nlmode = fulldisplay AND NOT newfile THEN                 01063
               BEGIN                                                     01064
               recred();                                                 01065
               END;                                                      01066
      END;                                                               01067
%restore the state of the world%                                         01068
   rubabt _ savrub;  % restore state of rubout abort %                   01069
```

```
      rubmrk _ savmrk;                                                    01070
      delda(&da); %get rid of dummy da%                                   01071
      IF newfile THEN closeu(stid.stfile := 0)                            01072
      ELSE cupdfil(stid.stfile :=0, newversion, 0);                       01073
   %Now close any files which got left open by error%                     01074
      %We Don't want to lose primary output stuff, so restore to tty,
      and re-open immediately after close all files%                      01075
         IF modeflg # 6 THEN                                              01076
            BEGIN                                                         01077
            !gpjfn(4B5);                                                  01078
            r2.RH _ r2.LH;                                                01079
            !spjfn(4B5);                                                  01080
            IF NOT SKIP !closf(pmyjfn) THEN NULL;                         01081
            END;                                                          01082
   %Now expunge if flag set%                                              01083
      IF flagut(5, $tstfg) THEN                                           01084
         BEGIN                                                            01085
         !JSYS gjinf;                                                     01086
         r1 _ r2; %connected directory number%                            01087
         !JSYS deldf; %expunge files%                                     01088
         END;                                                             01089
   %Now kill self if running detatched%                                   01090
      IF modeflg = 4  THEN                                                01091
         BEGIN                                                            01092
         clogout();     %go undo group stuff%                             01093
         IF NOT SKIP !closf(-1) THEN NULL; %close any openfiles%          01094
         IF NOT SKIP !lgout() THEN NULL; %logout%                         01095
         !haltf();                                                        01096
         END                                                              01097
      ELSE                                                                01098
         BEGIN                                                            01099
         crlf();                                                          01100
         typeas($"*** Tasks Completed ***");                              01101
         RETURN;                                                          01102
         END;                                                             01103
   END.
                                                                          01104
(setpriority)PROC(qcode);                                                  0783
   LOCAL pcap;                                                             0784
   IF (pcap _ enablw()) = -1 THEN RETURN;                                  0785
   r1 _ 4B5;                                                               0786
   r2 _ qcode;                                                             0787
   !JSYS 243B;                                                             0788
   IF pcap # -1 THEN disablw(pcap);                                        0789
   RETURN;                                                                 0790
   END.                                                                    0791
(setupop) PROC (jfn, filnamstr, modeflg, tty);                             0587
   %This procedure openas a file with the name in filnamstr if jfn = 0,
   otherwise uses file identified by jfn%                                  0588
   %It returns the JFN of the new Primary output file%                     0589
   %Types out a primary outfut file messae to tty indicated (-1 means
   controling)%                                                            0590
   %Detatches if modeflg = 4%                                              0591
   LOCAL STRING filnmstr50];                                               0592
   REF filnamstr;                                                          0593
   %Open a sequential file for primary output%                             0594
```

```
   IF jfn = 0 THEN                                                    0595
      BEGIN                                                           0596
      *filnms* _ *filnamstr*, EOL;                                    0597
      IF NOT (jfn _ sgtjfn(getgtjflgs(write, 0, dfltvrs), $filnms,
      $lit))                                                          0598
      OR NOT sysopen(jfn, write, chrtyp, $lit)                        0599
      OR NOT sysclose(jfn .V 4B11)                                    0600
      OR NOT sysopen(jfn, write, chrtyp, $lit) THEN err($lit);        0601
      %Type out file version number%                                  0602
         jfnstr(jfn, $lit);                                           0603
         *lit* _ "Primary Output File is ", *lit*;                    0604
         IF tty # -1 THEN                                             0605
            specttyout(tty, $lit)                                     0606
         ELSE                                                         0607
            BEGIN                                                     0608
            crlf();                                                   0609
            typeas($lit);                                             0610
            END;                                                      0611
      r1 _ jfn;                                                       0612
      r2 _ 3;                                                         0613
      IF NOT SKIP !JSYS delnf THEN NULL;                              0614
      END;                                                            0615
   %Now make it the primary output file%                              0623
      r1 _ 4B5;                                                       0624
      !gpjfn(); %get primary JFN's%                                   0625
      r2.RH _ jfn;                                                    0626
      r1 _ 4B5;                                                       0627
      !spjfn(); %set Primary JFN's%                                   0628
   %Type out detatching message and detach%                           0616
      IF (modeflg = 4) AND tty = -1 THEN                              0617
         BEGIN                                                        0618
         crlf();                                                      0619
         typeas($"Detatching");                                       0620
         crlf();                                                      0780
         !dtach();                                                    0621
         END;                                                         0622
   RETURN(jfn);                                                       0629
   END.                                                               0630
(runcmp) PROC (fnmstr, da);                                           0631
   LOCAL stid;                                                        0632
   LOCAL TEXT POINTER z1, z2, z3, z4;                                 0633
   LOCAL STRING cmpnam[50], rfilnm[50], lookst[5]; %For rel-file name%
                                                                      0634
   %This procedur accepts the name of a source code file, and compiles
   it with the compiler indicated be first satemen in e file, to te
   indicated rel-file.  If anything in the firs statement does not match
   the accepted syntax ( [percent] $NP COMPILER [percent] < $NP $LD >
   FILE percent) THen if the first statement is te FILE statement of an
   L10 Program, the file is compiled as L10 to the relfile under REL-NLS
   with the same name as te FILE.                                     0635
   %                                                                  0636
   REF fnmstr, da;                                                    0637
   stid _ orgstid;                                                    0638
   ON SIGNAL                                                          0639
      =ofilerr:                                                       0640
         BEGIN                                                        01110
```

```
      BUMP tskerrcnt;                                                01111
      err($"Rel-File open fail");                                    01106
      END;                                                           01112
   =prcerr:                                                           0641
      BEGIN                                                          01107
      BUMP tskerrcnt;                                                01109
      err($"Compiler open fail or no such compiler");                01105
      END;                                                           01108
   ELSE                                                               0642
      BEGIN %error%                                                   0643
      BUMP tskerrcnt;                                                01113
      IF stid.stfile THEN close(stid.stfile  := 0);                   0644
      END;                                                            0645
*tnmstr* _ *fnmstr*, ".NLS",  EOL;                                   0646
IF NOT (stid.stfile _ rawopen(&fnmstr, 0)) THEN                       0647
   err($"Cannot Open Source File");; %Open Source code file%        01114
makeptr(stid, $z1);                                                   0648
*lookst* _ "FILE";                                                    0649
lookup($z1, $lookst, contnt);                                         0650
IF  z1 = endfil THEN err($"Bad Source Code File");                    0651
stid _ z1;                                                            0652
%Get rel-file name and compiler%                                      0653
   CCPOS SF(stid);                                                    0654
   IF NOT FIND ['%] $NP ^z1 $PT ^z2 ['%] < CH $NP ^z4 [NP] > CH ^z3
   THEN                                                               0655
      IF NOT FIND  $NP "FILE" $NP ^z1 $LD ^z2 THEN                    0656
         err($"Illegal Format Source File")                           0657
      ELSE                                                            0658
         BEGIN                                                        0659
         *rfilnm* _ "<REL-NLS>", z1 z2, ".REL", EOL;                  0660
         *cmpnam* _  "L10";                                           0661
         END                                                          0662
   ELSE                                                               0663
      BEGIN                                                           0664
      *rfilnm* _  z3 z4, EOL;                                         0665
      *cmpnam* _  z1 z2;                                              0666
      END;                                                            0667
%call compiler%                                                       0668
   da.dacsp _ stid;                                                   0669
   IF (tskerrcnt _ cpcmpfl(TRUE, $cmpnam, $rfilnm, &da)) > 0 THEN
                                                                      0670
      BEGIN                                                           0671
      *lit* _  STRING(tskerrcnt), " Errors";                          0672
      err($lit);                                                      0673
      END;                                                            0674
%Close source file%                                                   0675
   close(stid.stfile := 0);                                           0676
%Return normally%                                                     0677
RETURN;                                                               0678
END.

                                                                      0679
(outptr) PROC (fnmstr, da, copies, lpt);                              0680
   LOCAL stid, pfjfn;                                                 0681
   LOCAL TEXT POINTER z1, z2;                                         0682
   LOCAL STRING  pfilnm[50], filnmstring[50]; %For rel-file name%     0683
```

```
   % This procedure accepts address of astring containing a file name
   and does an output quickprint on the file.  It sets the extension to
   be the string equivalent of copies.  If lpt is TRUE, then the file is
   put into the printers directory, otherwise in the connected
   directory. %                                                          0684
   REF fnmstr,da;                                                        0685
   stid _ orgstid;                                                       0686
   ON SIGNAL ELSE                                                        0687
      BEGIN                                                              0688
      IF stid.stfile THEN close (stid.stfile  := 0);                     0689
      IF lptjfn THEN sysclose (lptjfn  := 0);                            0690
      END;                                                               0691
   *filnmstring* _ *fnmstr*, ".NLS",  EOL;                               0692
   IF NOT (stid.stfile _ rawopen ($filnmstring, 0)) THEN                 0693
      err ($"Cannot Open File"); %Open Source code file%                 0694
   %Get output file name%                                                0695
      *pfilnm* _ NULL;                                                   0696
      CCPOS SF(*filnmstring*);                                           0697
      FIND $NP  ^z1 SE(z1) ^z2 >  z1 ([">] ^z1/)  ["."] ^z2 _z2;         0698
      IF lpt THEN                                                        0699
         *pfilnm* _ "<, *prtdir*, "> ;                                   0700
      *pfilnm* _ *pfilnm*,  z1 z2, ".", STRING(copies), EOL;             0701
   %turn statment numbers on (on the right)%                             0702
      da.dacsp _ stid;                                                   0703
      da.davspec.vsstnr _ da.davspec.vsstnf _ da.davspec.vssidf _ TRUE;
                                                                         0704
   %call Output Quickprint%                                              0705
      coutgui ($pfilnm, &da);                                            0706
   %Close Source File%                                                   0707
      close(stid.stfile := 0);                                           0708
   %Return normally%                                                     0709
   RETURN;                                                               0710
   END.


                                                                         0711
(copsrc) PROC (fnmstr);                                                  0712
   % This procedure accepts address of astring containing a file name.
   It changes the directory name for the rel file (in the "FILE"
   statement) to NIC-NLS, and then does an output file to the NIC-NLS
   directory. %                                                          0713
   LOCAL stid;                                                           0714
   LOCAL TEXT POINTER ptr1, ptr2, z1, z2;                                0715
   LOCAL STRING  nfname[50], lookst[5];                                  0716
   REF fnmstr;                                                           0717
   stid _ orgstid;                                                       0718
   ON SIGNAL ELSE                                                        0719
      IF stid.stfile THEN close (stid.stfile  := 0);                     0720
   *nfname* _ *fnmstr*, ".NLS",  EOL;                                    0721
   IF NOT (stid.stfile _ rawopen ($nfname, 0)) THEN                      0722
      err ($"Cannot Open File"); %Open Source code file%                 0723
   makeptr(stid, $ptr1);                                                 0724
   *lookst* _ "FILE";                                                    0725
   lookup($ptr1, $lookst, contnt);                                       0726
   IF ptr1 # endfil THEN                                                 0727
      BEGIN %change directory name%                                      0728
      stid _ ptr1;                                                       0729
```

```
   CCPOS SF(stid);                                                          0730
   IF FIND 2[~%] < [~>] ^ptr2 [~<] > CH ^ptr1 THEN                          0731
      ST stid _ SF(stid) ptr1, "NIC-NLS", ptr2 SE(stid);                    0732
   END;                                                                     0733
%make output file name%                                                     0734
   IF *fnmstr*[1] = ~< THEN                                                 0735
      BEGIN                                                                 0736
      CCPOS SF(*fnmstr*);                                                   0737
      IF NOT FIND [~>] ^ptr1 THEN                                           0738
         err($"Illegal file name");                                         0739
      *nfname* _ "<NIC-NLS>", ptr1 SE(*fnmstr*), ".NLS", EOL;               0740
      END                                                                   0741
   ELSE *nfname* _ "<NIC-NLS>", *fnmstr*, ".NLS", EOL;                      0742
%now do output and clean up%                                                0743
   cupdfil(stid.stfile, newname, $nfname);                                  0744
   %Use nfname as work string for transdir%                                 0745
   *nfname* _ "NIC-NLS";                                                    0746
   FIND SF(*nfname*) ^z1 SE(*nfname*) ^z2;                                  0747
   csetlindef(lcfile(), $z1, $z2);                                          0748
   close(stid.stfile := 0);                                                 0749
RETURN;                                                                     0750
END.

                                                                            0751
                                                                            0752
FINISH                                                                      0753
```