IDENT SUPPORT.

```
< NLS, IDENTSUPPORT.NLS;14, >, 15-JUN-77 10:51 KJM ;;;;( NLS,
CIDENT.NLS;12, ), 23-MAY-74 13:54 HGL ;
FILE identsupport % L10 to <rel-NLS>identsupport %% (L10,)
(rel-nls,identsupport.rel,) %                                          02
%....declarations....%                                                 03
   DECLARE EXTERNAL lname=1, strinname=2, idchr=3, afgptyp=4;          06
   DECLARE EXTERNAL STRING nwidstr = "NEWIDS", nullfield = "NONE";  02781
%.....load and nail down idents.master file.....%                     02780
   (loaidfil) PROCEDURE;  %load the Master Ident File Read Only%     02773
      LOCAL fileno;                                                    02774
      fileno _ open(0, jflname($"identfile"));                         02775
      [fintadr(fileno)].flnoclos _ TRUE;                               02776
      RETURN(fileno);                                                  02777
      END.                                                             02778
                                                                       02779
%....check idents against ident-file....%                              08
   (cknlsid) %validate ident for nls user%
   PROCEDURE (identsr, infostr, fileno);                               09
      %This procedure is used to validate idents for use in entering
      nls.  Currently it assumes that any ident that isn't a group or
      organization is legal%                                           010
      %-----------------%                                             011
      IF NOT infostr THEN                                              012
         IF NOT oldid(identsr, fileno) THEN RETURN(FALSE)              013
         ELSE NULL                                                     014
      ELSE                                                             015
         IF NOT ckident(identsr, infostr, fileno) THEN RETURN(FALSE);
                                                                       016
      IF orgrptst(identsr, 0) THEN RETURN(FALSE);                      017
      RETURN(TRUE);                                                    018
      END.
                                                                       019
   (ckident) %validate an ident and fetch entry if desired%
   PROCEDURE (identsr, retinfo, fileno);                            02660
      LOCAL idstid, retval, stid, count;                            02661
      LOCAL TEXT POINTER tp1, tp2;                                  02662
      REF identsr, retinfo, lgngrps;                                02663
      %This routine checks the validity of the ident in identsr, and
      returns the information concerning that person from  he systems
      ident file in the retinfo string.  If fileno is non-zero, it is
      assumed to be the file number of the ident file...otherwise, the
      ident file is opened%                                         02664
      %If a valid ident, also returns the stid of the statement which
      corresponds to the ident.%                                    02665
                                                                    02666
      IF identsr.L = empty THEN RETURN(FALSE);                      02667
      idstid _ orgstid;                                             02668
      IF NOT fileno THEN                                            02669
         BEGIN                                                      02670
         ON SIGNAL ELSE close(idstid.stfile := 0);                  02671
         idstid.stfile _ open(0, jflname($"identfile"));            02672
         END                                                        02673
      ELSE                                                          02674
         idstid.stfile _ fileno;                                    02675
      astruc(&identsr);                                             02676
      IF (retval _ ((stid _ namelook(idstid, &identsr)) # endfil)) THEN
```

```
                                                                    02677
    BEGIN                                                           02678
    IF NOT (FIND SF(stid) [')] $SP "LAST NAME") THEN                02679
       BEGIN                                                        02680
       IF &retinfo > 0 THEN                                         02681
          BEGIN                                                     02682
          *retinfo* _ SF(stid) SE(stid);                            02683
          IF NOT &lgngrps AND *identsr* = *initsr* THEN             02684
             BEGIN                                                  02685
             getigrps (&retinfo, 0, $tp1, $tp2);                    02686
             IF (count _ tp2 [1] - tp1 [1]) THEN                    02687
                BEGIN                                               02688
                &lgngrps _ getstring (count, $dspblk);              02689
                *lgngrps* _ tp1 tp2;                                02690
                END                                                 02691
             ELSE &lgngrps _ $" ";                                  02692
             END;                                                   02693
          END;                                                      02694
       END                                                          02695
    ELSE retval _ FALSE;                                            02696
    END;                                                            02697
 IF NOT fileno THEN close(idstid.stfile := 0);                      02698
 RETURN (retval, stid);                                             02699
 END.
                                                                    02700
(ckidlist) %check if list of idents are already being used%
PROCEDURE (idlist, badidlist, fileno);                               047
   %this routine assumes an identlist in IDLIST and returns with
   IDLIST containing a list of valid (in use) Idents and appends to
   BADIDLIST a list of those idents which are not now in use.%       048
   LOCAL startstid, idstid, stid;                                    049
   LOCAL TEXT POINTER z1, z2, z3, z4, c1, c2;                        050
   LOCAL STRING                                                      051
      identsr[50], work[50], comment[150], special[10],
      origlist[500];                                                02410
   REF idlist, badidlist;                                            052
   %-----------------%                                             02409
   IF idlist.L = empty THEN RETURN;                                  054
   IF badidlist.L THEN *badidlist* _ *badidlist*, SP;                053
   %If fileno is non-zero, it is assumed to be the file number of the
   ident file...otherwise, the ident file is opened%                 055
      idstid _ orgstid;                                              056
      IF NOT fileno THEN                                             057
         BEGIN                                                       058
         ON SIGNAL ELSE IF idstid.stfile THEN close(idstid.stfile :=
         0);                                                         059
         idstid.stfile _ open(0, jflname($"identfile"));             060
         END                                                         061
      ELSE                                                           062
         idstid.stfile _ fileno;                                     063
   *origlist* _ *idlist*;                                            064
   *idlist* _ NULL;                                                  065
   FIND SF(*origlist*) ^c2;                                          066
   IF (startstid _ namelook(idstid, $"'usedids'")) NOT = endfil AND
   (startstid := getsub(startstid)) NOT= startstid THEN              067
      LOOP  %process each ident in origidlist%                       068
```

```
        BEGIN                                                          069
        FIND c2 $(SP/",) ^z3 $1("^/"&) ^z4 ^z1 $(LD/"-) ^z2 $SP ("(
        ^c1 _c1 ["")] ^c2 / ^c1 ^c2);                                  070
        *identsr* _ +z1 z2;                                            071
        IF NOT identsr.L THEN EXIT; %no more idents%                   072
        *comment* _ c1 c2;                                            02404
        *special* _ z3 z4;                                            02407
        *work* _",, *identsr*, ",;                                     073
        stid _ startstid;                                              074
        LOOP  %scan the list of assigned idents to find this one%
                                                                       075
           BEGIN                                                       076
           FIND SF(stid) ^z1;                                          077
           WHILE ( FIND z1 [*identsr*] ^z1 ) DO                        078
              IF FIND < [",] > *work* THEN                             079
                 BEGIN                                                 080
                 *idlist* _ *idlist*, *special*, *identsr*,
                 *comment*, SP;                                        081
                 EXIT LOOP 2;                                          082
                 END;                                                  083
           IF getftl(stid) THEN                                        084
              BEGIN                                                    085
              *badidlist* _ *badidlist*, *special*, *identsr*,
              *comment*, SP;                                           086
              EXIT LOOP;                                               087
              END;                                                     088
           stid _ getsuc(stid);                                        089
           END;                                                        090
        END;                                                           091
   IF NOT fileno THEN close(idstid.stfile := 0);                       092
   IF idlist.L THEN BUMPDOWN idlist.L;                                02405
   IF badidlist.L THEN BUMPDOWN badidlist.L;                          02406
   RETURN;                                                             093
   END.                                                                094
                                                                      02408
(oldid) %check if an ident is already being used%
PROCEDURE (identsr, fileno);                                           095
   %this routine returns true if the ident passed is aready in use,
   false otherwise -- Much faster than ckident%                        096
   LOCAL idstid, stid;                                                 097
   LOCAL TEXT POINTER z1;                                              098
   LOCAL STRING work[200], ident[100];                                 099
   REF identsr;                                                       0100
   %If fileno is non-zero, it is assumed to be the file number of the
   ident file...otherwise, the ident file is opened%                  0101
      IF identsr.L = empty THEN RETURN(FALSE);                        0103
      idstid _ orgstid;                                               0102
      IF NOT fileno THEN                                              0104
         BEGIN                                                        0105
         ON SIGNAL ELSE close(idstid.stfile := 0);                    0106
         idstid.stfile _ open(0, jflname($"identfile"));              0107
         END                                                          0108
      ELSE                                                            0109
         idstid.stfile _ fileno;                                      0110
   *ident* _ *identsr*;                                               03252
   astruc(&identsr);                                                   0111
```

```
    *work* _ ',, *identsr*, ',;                                          0112
    IF (stid _ namelook(idstid, $"'usedids'")) NOT = endfil AND (stid
    := getsub(stid)) NOT= stid THEN                                      0113
       LOOP                                                              0114
          BEGIN                                                          0115
          FIND SF(stid) ^z1;                                             0116
          WHILE ( FIND z1 [*identsr*] ^z1 ) DO                           0117
             IF FIND < [',] > *work* THEN                                0118
                BEGIN                                                    0119
                IF NOT fileno THEN close(idstid.stfile := 0);            0120
                *identsr* _ *ident*;                                    03253
                RETURN(TRUE);                                            0121
                END;                                                     0122
          IF getftl(stid) THEN EXIT LOOP;                                0123
          stid _ getsuc(stid);                                           0124
          END;                                                           0125
    IF NOT fileno THEN close(idstid.stfile := 0);                        0126
    *identsr* _ *ident*;                                                03254
    RETURN(FALSE);                                                       0127
    END.                                                                 0128
%....Retrieve logical fields from ident-file entry....%                  0245
   (idelivery) %get LOGICAL DELIVERY TYPE(S) for an ident entry%
   PROCEDURE (stid);                                                     0246
      %stid points to identification record.                             0247
         Returns record of delivery types indicated in/by record (see
         record definition 'deliverymode' in utilty for possible types)%
                                                                         0248
      LOCAL retval, oldelf, arcorgf, astrng;                             0249
      LOCAL TEXT POINTER z1, z2, z3, z4;                                 0251
      LOCAL STRING tempsr[50];                                           0250
      REF astrng;                                                       02701
      &astrng _ 0;                                                      02704
      ON SIGNAL ELSE                                                    02722
         IF &astrng THEN freestring(&astrng:=0, $dspblk);               02723
      &astrng _ getstring(2000, $dspblk);                               02705
      IF NOT stid.stastr THEN                                            0253
         BEGIN                                                           0254
         *astrng* _ SF(stid) SE(stid);                                   0255
         stid _ asrref(&astrng);                                         0256
         END;                                                            0257
      getidelivery(stid.stadr, 0, $z1, $z2);                             0258
      retval _ 0;                                                        0259
      retval.delhc _ FIND BETWEEN z1 z2([*"Hardcopy"]/[*"Hard Copy"]);
                                                                         0260
      retval.delnet _ FIND BETWEEN z1 z2([*"Network"]/[*"Net Work"]);    0261
      oldelf _ FIND BETWEEN z1 z2([*"Online"]/[*"On-Line"]);             0262
      IF retval = 0 OR oldelf THEN                                       0263
         BEGIN                                                           0264
         getiorg(stid.stadr, $tempsr, 0, 0);                             0265
         arcorgf _ IF *tempsr* = "SRI-ARC" THEN TRUE ELSE FALSE;         0266
         END;                                                            0267
      IF retval = 0 AND NOT oldelf THEN                                  0268
         BEGIN                                                           0269
         %figure out a default delivery%                                 0270
         %NLS users and ARC members get On-Line%                         0271
            IF arcorgf THEN oldelf _ TRUE                                0272
```

```
         ELSE                                                          0273
            BEGIN                                                      0274
            %everyone gets hard copy now%                              0275
               retval.delhc _ TRUE;                                    0276
            geticapability(stid.stadr, 0, $z1, $z2);                   0277
            oldelf _ FIND BETWEEN z1 z2([*"NLS"*]);                    0278
            END;                                                       0279
      END;                                                             0280
   IF oldelf THEN                                                      0281
      BEGIN                                                            0282
      oldelf _ IF arcorgf THEN 2 ELSE 1; %set default nlhost%          0283
      getinlhost(stid.stadr, 0, $z3, $z4);                             0284
      IF z3[1] # z4[1] THEN                                            0285
         BEGIN                                                         0286
         oldelf _ 0;                                                   0287
         IF FIND BETWEEN z3 z4 ([*arcistr*]) THEN oldelf _ oldelf .V
         2;                                                            0288
         IF FIND BETWEEN z3 z4 ([*utilistr*]) THEN oldelf _ oldelf .V
         1;                                                            0289
         IF FIND BETWEEN z3 z4 ([*nsastr*]) THEN oldelf _ oldelf .V
         4;                                                           03331
         END;                                                          0290
      oldelf _ oldelf .A (CASE lhostn OF                               0291
         =archost: 2B;                                                 0292
         =utilhost: 1B;                                                0293
         =nsahost: 4B;                                                03332
         ENDCASE 0);                                                   0294
      IF oldelf THEN retval.delol _ TRUE;                              0295
      END;                                                             0296
   IF &astrng THEN freestring(&astrng:=0, $dspblk);                   02724
   RETURN(retval);                                                     0297
   END.
                                                                       0298
(luser) %get LOGICAL USER-NAME (for NLS delivery) for an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0299
   LOCAL TEXT POINTER lptr1, lptr2;                                    0300
   LOCAL retval;                                                       0301
   REF entrystr;                                                       0302
   retval _ FALSE;                                                     0303
   IF NOT                                                              0304
      (FIND SF(*entrystr*) [*"User:"*]                                 0305
         $NP  ^lptr1 [*";*] < CH $NP > ^lptr2) THEN                    0306
            getilname(&entrystr, fldstr, ptr1, ptr2)                   0307
   ELSE                                                                0308
      BEGIN                                                            0309
      stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                     0310
      retval _ TRUE;                                                   0311
      END;                                                             0312
   RETURN(retval);                                                     0313
   END.
                                                                       0314
(igetsubcoll) %get LOGICAL SUBCOLLECTIONS for an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0315
   LOCAL TEXT POINTER lptr1, lptr2;                                    0316
   LOCAL retval;                                                       0317
   LOCAL STRING tempsr[30];                                            0318
```

```
   REF entrystr;                                                    0319
   IF NOT (retval _ getisubcol(&entrystr, fldstr, ptr1, ptr2)) THEN
                                                                    0320
      IF orgrptst(&entrystr, 0) THEN getiid(&entrystr, fldstr, ptr1,
      ptr2)                                                         0321
      ELSE                                                          0322
         BEGIN                                                      0323
         getiorg(&entrystr, $tempsr, 0, 0);                         0324
         IF lhostn # nsahost THEN IF *tempsr* # "SRI-ARC" THEN
         *tempsr* _ "NIC";                                          0325
         FIND SF(*tempsr*) ^lptr1 SE(*tempsr*) > ^lptr2;            0326
         stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);               0327
         END;                                                       0328
   RETURN(retval);                                                  0329
   END.
                                                                    0330
(laddress) %get LOGICAL U.S. MAIL ADDRESS for an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2, fileno);                   0331
   % Fileno is the file number of an open file in which the address
   ident is checked;  it is passed to ckident.  If zero, the
   identfile is used and opened.  Entrystr is the address of a
   string containing the text of the ident entry whose mailing
   address is sought; fldstr is the address of a strig into which the
   mailing address is placed.  ptr1 and ptr2 are either the address
   of two text pointers which will delimit the  mailing address in
   the field string passed; they may be zero.  The procedure returns
   false if the address is not found. %                             0332
   %Get the mailing address, even if you have to go to another ident
   for it -- New Version by WLB 5 July 1972; modified by HGL
   18-DEC-73%                                                       0333
   LOCAL retval;                                                    0334
   LOCAL TEXT POINTER tpf, tpe;                                     0335
   LOCAL STRING savedid[30];                                        0336
   REF fldstr, entrystr, ptr1, ptr2;                                0337
   IF NOT &fldstr THEN RETURN (FALSE);                              0338
   getiadd(&entrystr, &fldstr, &ptr1, &ptr2);                       0339
   IF NOT FIND SF(*fldstr*) [NP] THEN                               0340
      BEGIN %See if it is an ident%                                 0341
      % save away this ident:  the entrystr will be smashed and then
      restored using it.  This is done to avoid using space.  It is
      slow and could be avoided by using another string. %          0342
      getiid (&entrystr, $savedid, 0, 0);                           0343
      IF ckident(&fldstr, &entrystr, fileno) THEN                   0344
         BEGIN % entrystr now has the info for the ident (a group or
         person) which has the address. %                           0345
         retval _ laddress(&entrystr, &fldstr, &ptr1, &ptr2, fileno);
                                                                    0346
         getinam (&entrystr, 0, $tpf, $tpe);                        0347
         IF orgtst (&entrystr, 0) THEN % Must have organization name
         at beginning %                                             0348
            *fldstr* _ tpf tpe, EOL, *fldstr*                       0349
         ELSE *fldstr* _ "c/o ", tpf tpe, EOL, *fldstr*;            0350
         % Now restore original string. %                           0351
         IF ckident ( $savedid, &entrystr, fileno ) AND retval THEN
                                                                    0352
            BEGIN                                                   0353
```

```
            IF &ptr1 THEN FIND SF(*fldstr*) ^ptr1;                    0354
            IF &ptr2 THEN FIND SE(*fldstr*) ^ptr2;                    0355
            RETURN(TRUE);                                             0356
            END                                                       0357
         ELSE RETURN( FALSE);                                         0358
         END;                                                         0359
      IF NOT ckident ( $savedid, &entrystr, fileno ) THEN RETURN
      (FALSE); % Return FALSE if we could not restore the original
      information %                                                   0360
      END;                                                            0361
   IF &ptr1 THEN FIND SF(*fldstr*) ^ptr1;                             0362
   IF &ptr2 THEN FIND SE(*fldstr*) ^ptr2;                             0363
   RETURN (TRUE);                                                     0364
   END.
                                                                      0365
(lmemlist) %get LOGICAL MEMBERSHIP for a group/organization ident%
PROCEDURE (entrystr, outlist, explist, noexplist, errorlist, idfile);
                                                                      0366
   % Gets the logical membership list of a group or organization --
   i.e., expands membership list (recursively) until only individual
   idents are present in list.  This routine is supposed to correctly
   handle expand/noexpand issues and also will not loop in the event
   of circular references -- e.g., GROUPA is a member of GROUPB is a
   member of GROUPC is a member of GROUPA -- isn't that a cute
   feature!                                                           0367
      ENTRYSTR is the name of a string containing the text of an
      identfile entry (as copied from the identfile) -- the name,
      coordinator, and memlist of the group/organization being
      referenced will be extracted from this string.                  0368
         NOTE: this must be a legitimate L10 STRING, as it will be
         used for working space by lmemlist -- the original contents
         will be restored before lmemlist returns, however.           0369
      OUTLIST is the name of a string in which the logically expanded
      memlist will be returned -- it must be large enough to hold the
      requested results!                                              0370
         In the output list each individual's ident is followed by a
         parenthetical expression containing a list of his
         "capacities" seperated by spaces.                            0371
         A person's capacity is the ident of a group or organization
         in whose memlist his ident appears.                          0372
            E.g., if GROUPC is a member of GROUPB, GROUPB is a member
            of GROUPA, and JOEBLOW is a member of GROUPC, then
            JOEBLOW's capacity is GROUPC.                             0373
         If an individual has more than one capacity in a group, then
         all will be listed in the parenthetical expression following
         his ident in the output list -- i.e., his ident itself will
         appear only once.                                            0374
            E.g., if GROUPC is a member of GROUPB, GROUPB is a member
            of GROUPA, and JOEBLOW is a member of GROUPA and GROUPC,
            then JOEBLOW's part of OUTLIST would be:                  0375
               JOEBLOW(GROUPA GROUPC)                                 0376
         The first ident listed in OUTLIST will be that of the
         coordinator of the group/organization whose memlist is being
         expanded.                                                    0377
      EXPLIST is the name of a string in which will be returned a
      list of idents (each followed by a space) of all
```

```
    groups/organizations which have been expanded in the course of
    producing OUTLIST  -- i.e., if an ident appears as a capacity
    on OUTLIST it will appear as an entry on EXPLIST.          0378
    NOEXPLIST is the name of a string in which will be returned a
    list of idents (each followed by a space) of all
    groups/organizations which have been encountered but not
    expanded in the course of producing OUTLIST -- i.e., idents
    which were found preceeded by '& and idents for no-expand
    groups/organizations which were not found preceeded by '^. 0379
       NOTE that no group/organization which is found on NOEXPLIST
       will be found as a capacity on OUTLIST -- if a
       group/organization appears in both an expand and a no-expand
       context, the expand context takes precedent.            0380
    ERRORLIST is the name of a string in which will be returned a
    list of idents (each followed by a space) which were
    encountered in the course of producing OUTLIST and which could
    not be found in the identfile.                             0381
    IDFILE is the file number of the identfile, if open when
    lmemlist was called, or zero.                              0382
    %                                                          0383
LOCAL lmlflag, expchr;                                         0384
LOCAL TEXT POINTER p1, p2, p3, p4, inp, expp, noexpp;          0385
LOCAL STRING inlist[500], inid[20], capacity[20], idstr[20];   0386
REF entrystr, outlist, explist, noexplist, errorlist;          0387
IF NOT orgrptst (&entrystr,0) THEN RETURN(FALSE);              0388
% Initialize Group Scan %                                      0389
   getiid (&entrystr, 0, $p1, $p2); *inid* _ +p1 p2;           0390
   *explist* _ *inid*, SP;                                     0391
   FIND SF(*explist*) ^expp;                                   0392
   *outlist* _ NULL;                                           0393
   *noexplist* _ NULL;                                         0394
   *errorlist* _ NULL;                                         0395
   lmlflag _ TRUE;                                             0396
LOOP BEGIN                                                     0397
   IF NOT FIND expp > ^p1 [SP] ^expp ^p2 _p2 THEN EXIT LOOP;   0398
   *capacity* _ p1 p2;                                         0399
   IF lmlflag                                                  0400
      THEN lmlflag _ FALSE                                     0401
      ELSE IF NOT ckident ($capacity, &entrystr, idfile)       0402
         THEN BEGIN                                            0403
            *errorlist* _ *errorlist*, *capacity*, SP;         0404
            REPEAT LOOP;                                       0405
            END;                                               0406
   getimem (&entrystr, 0, $p1, $p2); *inlist* _ +p1 p2;        0407
   geticord (&entrystr, 0, $p1, $p2); *idstr* _ +p1 p2;        0408
   WHILE ( FIND SE(*inlist*) '; ) DO BUMPDOWN inlist.L;        0409
   IF idstr.L # 0 AND FIND SF(*inlist*) [ *idstr* ^p2 -(LD/'-) <
   p2 1$(LD/'-) $SP ^p1 > $SP *idstr* ]                        0410
      THEN *inlist* _ *idstr*, SP, SF(*inlist*) p1, p2
      SE(*inlist*)                                             0411
      ELSE *inlist* _ *idstr*, SP, *inlist*;                   0412
      % INLIST now has mem list with coordinator's id at front %
                                                               0413
   FIND SF(*inlist*) ^inp;                                     0414
   LOOP BEGIN                                                  0415
      FIND inp > $( $NP ^inp '( ['')] );                       0416
```

```
        expchr _ IF FIND inp '^ $NP ^inp THEN '^                          0417
           ELSE IF FIND inp '& $NP ^inp THEN '&                           0418
           ELSE 0;                                                        0419
        IF NOT FIND inp ^p1 L $(LD/'-) ^p2 ^inp THEN EXIT LOOP;           0420
        *idstr* _ p1 p2;                                                  0421
        IF NOT ckident ($idstr, &entrystr, idfile)                        0422
           THEN BEGIN                                                     0423
              *errorlist* _ *errorlist*, *idstr*, '(, *capacity*,
              '), SP;                                                     0424
              REPEAT LOOP;                                                0425
              END;                                                        0426
        IF orgrptst (&entrystr, 0)                                        0427
           THEN BEGIN                                                     0428
              IF FIND SF(*explist*) ( *idstr* SP / [ SP *idstr* SP ]
              )                                                           0429
                 THEN NULL                                                0430
                 ELSE CASE expchr OF                                      0431
                    = '^ : BEGIN                                          0432
                       *explist* _ *explist*, *idstr*, SP;                0433
                       END;                                               0434
                    = '& : IF NOT FIND SF(*noexplist*) ( *idstr* SP
                    / [ SP *idstr* SP ] )                                 0435
                       THEN *noexplist* _ *noexplist*, *idstr*, SP;
                                                                          0436
                    ENDCASE IF expdtst (&entrystr, 0)                     0437
                       THEN REPEAT CASE ('^)                              0438
                       ELSE REPEAT CASE ('&);                             0439
              END                                                         0440
           ELSE IF FIND SF(*outlist*) ( *idstr* '( / [ SP *idstr* '(
           ] ) ['),] ^p1 _p1                                              0441
              THEN *outlist* _                                            0442
                 SF(*outlist*) p1, SP, *capacity*, p1 SE(*outlist*)
                                                                          0443
              ELSE *outlist* _                                            0444
                 *outlist*, *idstr*, '(, *capacity*, '), SP;              0445
        END; %LOOP%                                                       0446
     END; %LOOP%                                                          0447
  FIND SF(*noexplist*) ^noexpp;                                           0448
  lmlflag _ FALSE;                                                        0449
  WHILE ( FIND noexpp ^p1 [SP] ^p2 _p2 ^noexpp) DO BEGIN                  0450
     *capacity* _ p1 p2;                                                  0451
     IF FIND SF(*explist*) ( *capacity* SP / [ SP *capacity* SP ] )
                                                                          0452
        THEN BEGIN                                                        0453
           FIND noexpp < ^p1 SP 1$(LD/'-) ^noexpp;                        0454
           *noexplist* _ SF(*noexplist*) noexpp, p1 SE(*noexpp*);
                                                                          0455
           END;                                                           0456
     END;                                                                 0457
  ckident ($inid, &entrystr, idfile); % Restore entrystr %                0458
  RETURN (TRUE);                                                          0459
  END.                                                                    0460
%....retrieve fields from entry in ident-file....%                        0461
  (getiid) %get IDENT field from an ident entry%
  PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0462
     LOCAL TEXT POINTER lptr1, lptr2;                                     0463
```

```
    REF entrystr;                                                    0464
    FIND SF(*entrystr*) $(SP/TAB) '( $NP ^lptr1 [']] <CH $NP> ^lptr2;
                                                                     0465
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                     0466
    RETURN (TRUE);                                                   0467
    END.
                                                                     0468
(getiadd) %get MAIL ADDRESS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0469
    LOCAL TEXT POINTER lptr1, lptr2;                                 0470
    REF entrystr;                                                    0471
    IF orgrptst(&entrystr, 0) THEN                                   0472
       FIND SF(*entrystr*) 4[EOL] $SP ^lptr1                         0473
    ELSE                                                             0474
       FIND SF(*entrystr*) 2[EOL] ^lptr1;                            0475
    IF NOT (FIND lptr1 [EOL EOL] < $NP ^lptr2 >) THEN                0476
       FIND lptr1 > ^lptr2;                                          0477
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                     0478
    RETURN (TRUE);                                                   0479
    END.
                                                                     0480
(getinam) %get NAME field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0481
    LOCAL TEXT POINTER lptr1, lptr2;                                 0482
    REF entrystr;                                                    0483
    IF orgrptst(&entrystr, 0) THEN                                   0484
       FIND SF(*entrystr*) 2[EOL] $SP ^lptr1                         0485
    ELSE                                                             0486
       FIND SF(*entrystr*) [EOL] $SP ^lptr1;                         0487
    FIND lptr1 [EOL] < $NP ^lptr2 >;                                 0488
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                     0489
    RETURN (TRUE);                                                   0490
    END.
                                                                     0491
(getitnf) %get NAME field (FIRST NAME FIRST) field from an individual
ident entry%
PROCEDURE (entrystr, fldstr); %first name first%                     0492
    LOCAL grpflg;                                                    0493
    LOCAL TEXT POINTER lptr1, lptr2, lptr3, lptr4;                   0494
    REF entrystr, fldstr;                                            0495
    IF (grpflg _ orgrptst(&entrystr, 0)) THEN                        0496
       FIND SF(*entrystr*) 2[EOL]                                    0497
    ELSE                                                             0498
       FIND SF(*entrystr*) [EOL];                                    0499
    FIND $SP ^lptr1 ^lptr2 ^lptr3 [EOL] < $NP ^lptr4 >;              0500
    IF NOT grpflg THEN FIND BETWEEN lptr1 lptr4 ([',] $NP ^lptr3 < $NP
    CH $NP ^lptr2);                                                  0501
    *fldstr* _ lptr3 lptr4, SP, lptr1 lptr2;                         0502
    RETURN (TRUE);                                                   0503
    END.
                                                                     0504
(getilname) %get LAST NAME field from an individual ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0505
    %Get last name%                                                  0506
    LOCAL TEXT POINTER lptr1, lptr2;                                 0507
    REF entrystr;                                                    0508
```

```
    getinam(&entrystr, 0, $lptr1, $lptr2);                              0509
    FIND lptr1 > [',] ^lptr2_lptr2;                                     0510
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0511
    RETURN (TRUE);                                                      0512
    END.
                                                                        0513
(getiorg) %get ORGANIZATION field from an individual ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               03073
    LOCAL TEXT POINTER lptr1, lptr2;                                    03074
    REF entrystr;                                                       03075
    IF orgrptst(&entrystr, 0)                                           03076
    OR NOT                                                              03077
       (FIND SF(*entrystr*) [')] $SP ^lptr1 [';/EOL] < CH $NP> ^lptr2
       ) THEN                                                           03078
          FIND SE(*entrystr*) ^lptr1 ^lptr2;                            03079
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        03080
    RETURN (TRUE);                                                      03081
    END.
                                                                        03082
(getiexp) %get EXPAND field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0524
    LOCAL TEXT POINTER lptr1, lptr2;                                    0525
    REF entrystr;                                                       0526
    FIND SF(*entrystr*) [')] $NP ^lptr1 ("Expand"/) ^lptr2;             0527
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0528
    RETURN (TRUE);                                                      0529
    END.
                                                                        0530
(getimem) %get MEMBERSHIP field from a group/organization ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0531
    LOCAL TEXT POINTER lptr1, lptr2;                                    0532
    REF entrystr;                                                       0533
    IF NOT orgrptst(&entrystr, 0)                                       0534
    OR NOT                                                              0535
       (FIND SF(*entrystr*) [EOL] $(SP/TAB) ^lptr1 PT [EOL] < $NP
       ^lptr2 >) THEN                                                   0536
          FIND SF(*entrystr*) [EOL] ^lptr1 ^lptr2;                      0537
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0538
    RETURN (TRUE);                                                      0539
    END.
                                                                        0540
(getiprevmem) %get MEMBERSHIPPREVIOUS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               02541
    LOCAL TEXT POINTER p1, p2;                                          02542
    LOCAL retval;                                                       02543
    REF entrystr;                                                       02544
    retval _ FALSE;                                                     02545
    IF NOT FIND SF(*entrystr*) ["MembershipPrevious:"] $NP ^p1 [';] <
    CH $NP > ^p2 THEN                                                   02546
       getiend (&entrystr, $p1, $p2)                                    02547
    ELSE retval _ TRUE;                                                 02548
    stptset (fldstr, ptr1, ptr2, $p1, $p2);                             02549
    RETURN (retval);                                                    02550
    END.                                                                02551
                                                                        02552
```

```
(geticord) %get COORDINATOR field from a group/organization ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0541
   LOCAL TEXT POINTER lptr1, lptr2;                                    0542
   REF entrystr;                                                       0543
   IF NOT orgrptst(&entrystr, 0)                                       0544
   OR NOT                                                              0545
      (FIND SF(*entrystr*) 3[EOL] $(SP/TAB) ^lptr1 PT                  0546
         [EOL] <$NP> ^lptr2 ) THEN                                     0547
            FIND SF(*entrystr*) 3[EOL] ^lptr1 ^lptr2;                  0548
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0549
   RETURN (TRUE);                                                      0550
   END.
                                                                       0551
(getityp) %get TYPE-OF-ORGANIZATION field from an organization ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0552
   LOCAL TEXT POINTER lptr1, lptr2;                                    0553
   REF entrystr;                                                       0554
   IF NOT orgrptst(&entrystr, 0)                                       0555
   OR NOT                                                              0556
      (FIND SF(*entrystr*) ["Type:"] $NP ^lptr1 [";] < CH $NP >
      ^lptr2 ) THEN                                                    0557
         getiend(&entrystr, $lptr1, $lptr2);                           0558
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0559
   RETURN (TRUE);                                                      0560
   END.
                                                                       0561
(getiverify) %get VERIFIED-BY-NIC-PERSONNEL field from an ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0562
   LOCAL TEXT POINTER lptr1, lptr2;                                    0563
   REF entrystr;                                                       0564
   IF NOT                                                              0565
      (FIND SF(*entrystr*) ["Verified"/"Unverified"] ^lptr2            0566
         < [EOL] > $NP ^lptr1) THEN                                    0567
            getiend(&entrystr, $lptr1, $lptr2);                        0568
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0569
   RETURN (TRUE);                                                      0570
   END.
                                                                       0571
(getiuser) %get USER (NLS delivery) field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0572
   LOCAL TEXT POINTER lptr1, lptr2;                                    0573
   LOCAL retval;                                                       0574
   REF entrystr;                                                       0575
   retval _ FALSE;                                                     0576
   IF NOT                                                              0577
      (FIND SF(*entrystr*) ["User:"]                                   0578
         $NP  ^lptr1 [";] < CH $NP > ^lptr2) THEN                      0579
            getiend(&entrystr, $lptr1, $lptr2)                         0580
   ELSE                                                                0581
      retval _ TRUE;                                                   0582
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0583
   RETURN(retval);                                                     0584
   END.
```

```
                                                                    0585
(getigrps) %get GROUPS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                           0586
   LOCAL strptr;                                                   03333
   LOCAL TEXT POINTER lptr1, lptr2;                                 0587
   LOCAL retval;                                                    0588
   REF entrystr;                                                    0589
   strptr _ getpointer(&entrystr);                                 03334
   retval _ FALSE;                                                  0590
   IF NOT FIND SF(strptr) ["Groups:"] $NP ^lptr1 [";] < CH $NP >
   ^lptr2 THEN                                                      0591
      getiend (&entrystr, $lptr1, $lptr2)                           0592
   ELSE retval _ TRUE;                                              0593
   stptset (fldstr, ptr1, ptr2, $lptr1, $lptr2);                    0594
   RETURN (retval);                                                 0595
   END.                                                             0596
                                                                    0597
(getpointer) %convert possible string address to a-string, leave
stids alone%
PROCEDURE (ptr);                                                    03335
   %handle stid, a-string, or address of string%                    03339
   IF NOT ptr.LH THEN ptr _ asrref(ptr);                            03336
   RETURN(ptr);                                                     03337
   END.
                                                                    03338
(getiphone) %get PHONE field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                           0598
   LOCAL TEXT POINTER lptr1, lptr2;                                 0599
   LOCAL retval;                                                    0600
   REF entrystr;                                                    0601
   retval _ FALSE;                                                  0602
   IF NOT                                                           0603
      (FIND SF(*entrystr*) ["Phone:"] $NP ^lptr1 [";] < CH $NP >    0604
         ^lptr2) THEN                                               0605
            getiend(&entrystr, $lptr1, $lptr2)                      0606
   ELSE retval _ TRUE;                                              0607
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                     0608
   RETURN(retval);                                                  0609
   END.
                                                                    0610
(getihost) %get NETWORK-DELIVERY HOST field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                           0611
   LOCAL TEXT POINTER lptr1, lptr2;                                 0612
   LOCAL retval;                                                    0613
   REF entrystr;                                                    0614
   retval _ FALSE;                                                  0615
   IF NOT                                                           0616
      (FIND SF(*entrystr*) ["Host:"] $NP ^lptr1 [";] < CH $NP >     0617
         ^lptr2) THEN                                               0618
            getiend(&entrystr, $lptr1, $lptr2)                      0619
   ELSE retval _ TRUE;                                              0620
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                     0621
   RETURN(retval);                                                  0622
   END.
                                                                    0623
(getinlhost) %get NLS-DELIVERY HOST field from an ident entry%
```

```
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0624
   LOCAL TEXT POINTER lptr1, lptr2;                                    0625
   LOCAL retval;                                                       0626
   REF entrystr;                                                       0627
   retval _ FALSE;                                                     0628
   IF NOT                                                              0629
      (FIND SF(*entrystr*) [^"NLS host:"] $NP ^lptr1 [^;] < CH $NP >
                                                                       0630
         ^lptr2) THEN                                                  0631
            getiend(&entrystr, $lptr1, $lptr2)                         0632
   ELSE retval _ TRUE;                                                 0633
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0634
   RETURN(retval);                                                     0635
   END.
                                                                       0636
(getinma) %get NETWORK MAILBOX Address field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0637
   LOCAL TEXT POINTER lptr1, lptr2;                                    0638
   LOCAL retval;                                                       0639
   REF entrystr;                                                       0640
   retval _ FALSE;                                                     0641
   IF NOT                                                              0642
      (FIND SF(*entrystr*) [^"Local Network Mailbox Address:"] $NP
      ^lptr1 [^; NP] < 2CH $NP >                                       0643
         ^lptr2) THEN                                                  0644
            getiend(&entrystr, $lptr1, $lptr2)                         0645
   ELSE retval _ TRUE;                                                 0646
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0647
   RETURN(retval);                                                     0648
   END.
                                                                       0649
(getifunction) %get FUNCTION field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0650
   LOCAL TEXT POINTER lptr1, lptr2;                                    0651
   LOCAL retval;                                                       0652
   REF entrystr;                                                       0653
   retval _ FALSE;                                                     0654
   IF NOT                                                              0655
      (FIND SF(*entrystr*) [^"Function:"] $NP ^lptr1 [^;] < CH $NP >
                                                                       0656
         ^lptr2) THEN                                                  0657
            getiend(&entrystr, $lptr1, $lptr2)                         0658
   ELSE retval _ TRUE;                                                 0659
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0660
   RETURN(retval);                                                     0661
   END.
                                                                       0662
(getisorg) %get SECONDARY ORGANIZATION field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0663
   LOCAL TEXT POINTER lptr1, lptr2;                                    0664
   LOCAL retval;                                                       0665
   REF entrystr;                                                       0666
   retval _ FALSE;                                                     0667
   IF NOT                                                              0668
      (FIND SF(*entrystr*) [^"Secondary organization:"] $NP ^lptr1
      [^;] < CH $NP > ^lptr2 ) THEN                                    0669
```

```
        getiend(&entrystr, $lptr1, $lptr2)                              0670
    ELSE retval _ TRUE;                                                 0671
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0672
    RETURN(retval);                                                     0673
    END.
                                                                        0674
(geticapability) %get CAPABILITIES field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0675
    LOCAL retval;                                                       0676
    LOCAL TEXT POINTER lptr1, lptr2;                                    0677
    REF entrystr;                                                       0678
    retval _ FALSE;                                                     0679
    IF NOT                                                              0680
       (FIND SF(*entrystr*) [~"Capabilities:"] $NP ^lptr1 [~;] < CH $NP
       >
                                                                        0681
          ^lptr2) THEN                                                  0682
             getiend(&entrystr, $lptr1, $lptr2)                         0683
    ELSE retval _ TRUE;                                                 0684
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0685
    RETURN(retval);                                                     0686
    END.
                                                                        0687
(getisubcol) %get SUBCOLLECTIONS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0688
    LOCAL TEXT POINTER lptr1, lptr2;                                    0689
    LOCAL retval;                                                       0690
    REF entrystr;                                                       0691
    retval _ FALSE;                                                     0692
    IF NOT                                                              0693
       (FIND SF(*entrystr*) [~"Sub-Collection:"] $NP ^lptr1 [~;] < CH
       $NP >
                                                                        0694
          ^lptr2 ) THEN                                                 0695
             getiend(&entrystr, $lptr1, $lptr2)                         0696
    ELSE retval _ TRUE;                                                 0697
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0698
    RETURN(retval);                                                     0699
    END.
                                                                        0700
(getidelivery) %get DELIVERY-TYPES field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0701
    LOCAL TEXT POINTER lptr1, lptr2;                                    0702
    LOCAL retval;                                                       0703
    REF entrystr;                                                       0704
    retval _ FALSE;                                                     0705
    IF NOT                                                              0706
       (FIND SF(*entrystr*) [~"Delivery:"] $NP ^lptr1 [~;] < CH $NP >
                                                                        0707
          ^lptr2 ) THEN                                                 0708
             getiend(&entrystr, $lptr1, $lptr2)                         0709
    ELSE retval _ TRUE;                                                 0710
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0711
    RETURN(retval);                                                     0712
    END.
                                                                        0713
(getimcmnts) %get COMMENTS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0714
```

```
    LOCAL TEXT POINTER lptr1, lptr2;                                    0715
    LOCAL retval;                                                       0716
    REF entrystr;                                                       0717
    retval _ FALSE;                                                     0718
    IF NOT                                                              0719
       (FIND SF(*entrystr*) ["Comments:"] $SF ^lptr1 [EOL] ^lptr2
       _lptr2)                                                          0720
          THEN FIND SE(*entrystr*) ^lptr1 ^lptr2                        0721
    ELSE retval _ TRUE;                                                 0722
    stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0723
    RETURN(retval);                                                     0724
    END.
                                                                        0725
%....support routines for get/set entry in ident-file....%              0726
  (getiend) %find the end of an ident entry%
  PROCEDURE (entrystr, ptr1, ptr2);                                     0727
    %fixed so entrystr may also be a-string or stid%                   03342
    LOCAL strptr;                                                      03340
    REF entrystr, ptr1, ptr2;                                           0728
    strptr _ getpointer(&entrystr);                                    03341
    IF NOT                                                              0729
       (FIND SF(strptr)                                                 0730
          ["Comments:"] EOL < 10 CH > ^ptr1 ^ptr2) THEN                 0731
            FIND SE(strptr) ^ptr1 ^ptr2;                                0732
    RETURN;                                                             0733
    END.
                                                                        0734
  (stptset) %set string and/or text-pointers for get routines%
  PROCEDURE (string, nptr1, nptr2, optr1, optr2);                       0735
    REF string, nptr1, nptr2, optr1, optr2;                             0736
    IF &string THEN *string* _ optr1 optr2;                             0737
    IF &nptr1 THEN                                                      0738
       FIND optr1 ^nptr1;                                               0739
    IF &nptr2 THEN                                                      0740
       FIND optr2 ^nptr2;                                               0741
    RETURN;                                                             0742
    END.
                                                                        0743
  (expdtst) %test expand field and set text-pointer to membership list%
  PROCEDURE (string, dstptr);                                           0744
    %This routine looks at the statement indicated by srcptr, and
    returns true or false to indicate whether the expand parameter is
    set.  In addition, if the second argument is non-zero, it assumes
    that this is the address of a t-pointr and updates the pointer to
    point to the beginning of the membership list.  If the membership
    list is not present, then the t-pointer will contain endchr.%       0745
    %--------------------%                                              0746
    REF string, dstptr;                                                 0747
    IF FIND SF(*string*) '( $(LD/''/'-) ') $(SP/TAB)                    0748
       "Expand" $(SP/TAB)                                               0749
       ("Group"/"Organization") $(SP/TAB) EOL                           0750
       THEN                                                             0751
          BEGIN %expand parameter set%                                  0752
          IF &dstptr THEN %set dstptr to beginning of membership list%
                                                                        0753
             IF FIND ^dstptr $(SP/TAB) THEN                             0754
```

```
            FIND ^dstptr;                                          0755
         RETURN(TRUE);                                             0756
         END                                                       0757
   ELSE RETURN(FALSE);                                             0758
   END.
                                                                   0759
%....test type of entry in ident-file....%                         0760
   (jgrptst) %test if a group entry%
   PROCEDURE (string, dstptr);                                     0761
      %This routine looks at the statement indicated by srcptr, and
      returns true or false to indicate whether it contains a group
      identification.  In addition, if the second argument is non-zero,
      it assumes that this is the address of a t-pointr and updates the
      pointer to point to the beginning of the membership list.%   0762
      %--------------------%                                       0763
      REF string, dstptr;                                          0764
      IF FIND SF(*string*) [']] $(SP/TAB) %statement name%         0765
         (("Expand" $(SP/TAB) "Group")/"Group")                    0766
         $(SP/TAB) EOL THEN                                        0767
            BEGIN %group id present%                               0768
            IF &dstptr THEN %set dstptr to beginning of membership list%
                                                                   0769
               IF FIND ^dstptr $(SP/TAB) THEN                      0770
                  FIND ^dstptr;                                    0771
            RETURN(TRUE);                                          0772
            END                                                    0773
      ELSE RETURN(FALSE);                                          0774
      END.
                                                                   0775
   (orgtst) %test if a organization entry%
   PROCEDURE (string, dstptr);                                     0776
      %This routine looks at the statement indicated by srcptr, and
      returns true or false to indicate whether it is an organization.
      In addition, if the second argument is non-zero, it assumes that
      this is the address of a t-pointer and updates the pointer to
      point to the beginning of the membership list.  If the membership
      list is not present, then the t-pointer will contain endchr.% 0777
      %--------------------%                                       0778
      REF string, dstptr;                                          0779
      IF FIND SF(*string*) [']] $(SP/TAB) %satatement name%        0780
         (("Expand" $(SP/TAB) "Organization")/"Organization")      0781
         $(SP/TAB) EOL  THEN                                       0782
            BEGIN %organization id present%                        0783
            IF &dstptr THEN %set dstptr to beginning of membership list%
                                                                   0784
               IF FIND ^dstptr $(SP/TAB) THEN                      0785
                  FIND ^dstptr;                                    0786
            RETURN(TRUE);                                          0787
            END                                                    0788
      ELSE RETURN(FALSE);                                          0789
      END.
                                                                   0790
   (orgrptst) %test if a group or organization entry%
   PROCEDURE (string, dstptr);                                     0791
      %This routine looks at string, and returns true or false to
      indicate whether it is an organization.  In addition, if the
```

```
    second argument is non-zero, it assumes that this is the address
    of a t-pointer and updates the pointer to point to the beginning
    of the membership list.  If the membership list is not present,
    then the t-pointer will contain endchr.%                          0792
    %--------------------%                                            0793
    %fixed to also work with a-string or stid%                        03343
    LOCAL strptr;                                                     03344
    REF string, dstptr;                                               0795
    strptr _ getpointer(&string);                                     03345
    IF FIND SF(strptr) [^)] $(SP/TAB) THEN                            0796
       BEGIN %statement name%                                         0797
       FIND "Expand" $(SP/TAB); %skip "Expand" if it occurs%          0798
       IF FIND ("Group"/"Organization") [EOL] THEN                    0800
          BEGIN %organization/group present%                          0801
          IF &dstptr THEN %set dstptr to beginning of membership list%
                                                                      0802
             FIND $(SP/TAB) ^dstptr;                                  0803
          RETURN(TRUE);                                               0805
          END;                                                        0806
       END;                                                           0807
    RETURN(FALSE);                                                    0808
    END.
                                                                      0809
%....expand list of idents into list of individuals....%              02841
   (getids) %expand list of idents into list of individuals%
   PROCEDURE (ptr, astr, infotype, idfile);                           02842
      LOCAL expchr, gpstid;                                           02843
      LOCAL TEXT POINTER idf, ide, tmpptr, srcptr, dstptr;            02844
      LOCAL STRING idstr[20], infostr[2000];                          02845
      REF ptr, astr;                                                  02846
      %reads an ident from pointer ptr into astring idstr, and then
      calls ckident to get info on it.                                02847
         IF infotype = 0, then it returns all of the info in astr, if
         infotype = 1, then the name only is returned.                02848
         Uses infostr as a work area%                                 02849
      expchr _ 0;                                                     02850
      %first, read ident%                                             02851
         LOOP                                                         02852
            BEGIN                                                     02853
            CCPOS ptr;                                                02854
            IF FIND $NP ^idf ": THEN                                  02855
               BEGIN                                                  02856
               IF NOT popids(&ptr) THEN RETURN(FALSE); %no more idents%
                                                                      02857
               END                                                    02858
            ELSE EXIT LOOP;                                           02859
            END;                                                      02860
         IF NOT FIND idf [ NP / ": / "( ]                             02861
            ^ptr _ptr ^ide _ide THEN                                  02862
               err($"Ident List Format Error");                       02863
         FIND ptr ($NP "( [^)] ^ptr);                                 02864
         IF FIND idf ("&/"^) ^idf  < CH > THEN expchr _ READC;        02865
         *idstr* _ idf ide; %ident%                                   02866
      %Now get info, and check ident%                                 02867
         IF ckident($idstr, $infostr, idfile : gpstid) THEN %return
         something%                                                   02868
```

```
            BEGIN                                                   02869
            IF orgrptst($infostr, 0) THEN                           02870
               BEGIN                                                02871
               expchr _ CASE expchr OF %expand group list or not%   02872
                  ="&: FALSE;                                       02873
                  ="^: TRUE;                                        02874
                  ENDCASE expdtst($infostr, 0); %take default from ident
                  record%                                           02875
               IF expchr THEN                                       02876
                  BEGIN                                             02877
                  getimem($infostr, 0,  $dstptr, 0);                02878
                  IF FIND dstptr -EOL %membership list present% THEN
                                                                    02879
                     BEGIN                                          02880
                     pushids(&ptr);                                 02881
                     dstptr _ gpstid;                               02882
                     FIND dstptr ^ptr;                              02883
                     RETURN(getids(&ptr, &astr, infotype, idfile));
                                                                    02884
                     END;                                           02885
                  END;                                              02886
               END;                                                 02887
            END;                                                    02888
         %Now edit and append to astr %                             02889
            IF infotype = 1 THEN                                    02890
               getifnf($infostr,  $infostr);                        02891
            *astr* _ *astr*, *infostr*;                             02892
      RETURN(TRUE) END.
                                                                    02893
%....ident pushdown stack support....%                              02815
   (intids) %initialize ident pushdown stack%
   PROCEDURE (ptr);                                                 02816
      RESET jidstk;                                                 02817
      IF ptr THEN pushids(ptr);                                     02818
      jidsbot _ jidstk;                                             02819
      RETURN;                                                       02820
      END.
                                                                    02821
   (popids) %pop the ident pushdown stack%
   PROCEDURE (ptr);                                                 02822
      REF ptr;                                                      02823
      IF jidsbot = jidstk THEN RETURN(FALSE);                       02824
      POP jidstk TO ptr;                                            02825
      RETURN(TRUE);                                                 02826
      END.
                                                                    02827
   (pushids) %push the ident pushdown stack%
   PROCEDURE (ptr);                                                 02828
      REF ptr;                                                      02829
      PUSH ptr ON jidstk;                                           02830
      RETURN;                                                       02831
      END.
                                                                    02832
   (gbotids) %collaps ident pushdown stack%
   PROCEDURE (ptr);                                                 02833
      REF ptr;                                                      02834
```

```
    IF jidsbot = jidstk THEN RETURN(FALSE);                          02835
    IF jidstk.systks=1 THEN ptr _ [$jidstk+2]                        02836
    ELSE mvbfbf($jidstk+2, &ptr, jidstk.systks);                     02837
    RETURN(TRUE);                                                    02838
    END.                                                             02839
                                                                     02840
%....miscellaneous utility routines....%                             01551
  (getgpids) %given an identlist, return list of only the group idents%
  PROCEDURE (ptr, astr, idfnum);                                     02782
    %Read identlist identified by ptr, and return all group idents
    referenced in the string astr, separated by spaces%              02783
    LOCAL TEXT POINTER z1, z2, z3;                                   02784
    LOCAL idfile;                                                    02785
    LOCAL STRING idstr[20], infostr[2000];                           02786
    REF ptr, astr;                                                   02787
    IF NOT idfnum THEN                                               02788
      BEGIN                                                          02789
      idfile _ 0;                                                    02790
      ON SIGNAL ELSE sigclose(idfile := 0);                          02791
      idfile _ open(0, jflname($"Identfile"));                       02792
      END                                                            02793
    ELSE idfile _ idfnum;                                            02794
    FIND ptr ^z1;                                                    02795
    WHILE (FIND z1 $NP ('^/'&/) ^z2 -'; [ NP / '; / '(] < CH > ^z3  (
    $NP '( ['')] / ) ^z1 ) DO                                        02796
      BEGIN                                                          02797
      *idstr* _ z2 z3;                                               02798
      IF NOT ckident($idstr, $infostr, idfile) THEN
      err($"Identification System Error");                           02799
      IF orgrptst($infostr, 0) THEN *astr* _ *astr*, SP, *idstr*;
                                                                     02800
      END;                                                           02801
    IF NOT idfnum THEN close(idfile := 0);                           02802
    RETURN;                                                          02803
    END.                                                             02804
  (stnamcap) %Capitalize first letter of each word in a string%
  PROCEDURE (string);                                                01552
    LOCAL char, count;                                               01553
    REF string;                                                      01554
    %-----------------%                                              01555
    count _ 1;                                                       01556
    CASE (char _ *string*[count]) OF                                 01557
      IN ['a, 'z]:  *string*[count] _ char - 40B;                    01558
      =SP:                                                           01559
        BEGIN                                                        01560
        *string* _ *string*[count+1 TO string.L];                    01561
        REPEAT CASE;                                                 01562
        END;                                                         01563
      ENDCASE;                                                       01564
    WHILE (count _ count + 1) < string.L                             01565
      DO                                                             01566
      IF *string*[count] = SP                                        01567
      AND (char _ *string*[count + 1]) IN ['a, 'z] THEN              01568
        BEGIN                                                        01569
        *string*[count + 1] _ char - 40B;                            01570
        count _ count + 1;                                           01571
```

```
        END;                                                    01572
    RETURN;                                                     01573
    END.
                                                                01574
(namesearch) %search and print routine%
PROCEDURE (string, idstr, type, fileno);                        03083
    %This procedure accepts a string containing a last name in namstr,
    and searches the identification file for entries with matching
    string -- last name, first character(s) of last name for
    individuals, or string in name...depending on type. When a match
    is found information is typed to the user, which is intended to
    identify the entry.  The number of hits is returned, plus if the
    number is greater than 0, idstr has the ident of the last one
    processed.%                                                 03084
                                                                03085
    LOCAL foundsome, retvalue,  savrubabt, stid, typid, idstid, i;
                                                                03086
    LOCAL TEXT POINTER tx1, tx2, tx3;                           03325
    LOCAL STRING work[1000], tempsr[200], uppercasestring[200],
    tbstr[1000];                                                03087
    REF idstr, string;                                          03088
    %Open id file%                                              03089
        idstid _ orgstid;                                       03090
        IF NOT fileno THEN                                      03091
            BEGIN                                               03092
            ON SIGNAL ELSE                                      03093
                sigclose(idstid.stfile := 0);                   03094
            idstid.stfile _ open(0, jflname($"identfile"));     03095
            END                                                 03096
        ELSE idstid.stfile _ fileno;                            03097
    savrubabt _ rubabt := FALSE;                                03098
    retvalue _ 0;                                               03099
    CASE type OF                                                03100
        = lname: %individual's last names%                      03101
            BEGIN                                               03102
            FOR i _ 1 UP UNTIL > string.L DO %remove leading spaces%
                                                                03103
                IF *string*[i] # SP THEN                        03104
                    BEGIN                                       03105
                    IF i NOT= 1 THEN                            03106
                        *string* _ *string*[i TO string.L];     03107
                    EXIT;                                       03108
                    END;                                        03109
            FOR i _ string.L DOWN UNTIL <= 0 DO %remove trailing spaces%
                                                                03110
                IF *string*[i] # SP THEN                        03111
                    BEGIN                                       03112
                    IF i NOT= string.L THEN                     03113
                        *string* _ *string*[1 TO i];            03114
                    EXIT;                                       03115
                    END;                                        03116
            *tempsr* _ "", *string*, "";                        03117
            FOR i _ 1 UP UNTIL > tempsr.L DO %convert spaces to "%
                                                                03118
                IF *tempsr*[i] = SP THEN *tempsr*[i] _ "";      03119
            IF (stid _ namelook(idstid, $tempsr)) = endfil OR NOT (FIND
```

```
      SF(stid) [")] $SP "LAST NAME") THEN                                  03120
         BEGIN                                                             03121
         *fbstr* _ CR, LF, "None", CR, LF;                                 03433
         fbctl(fbaddlit, $fbstr);                                          03122
         EXIT CASE;                                                        03123
         END;                                                              03124
      *fbstr* _ CR, LF, "The following individuals with last name
      ", *string*, " are already defined", CR, LF;                         03125
      fbctl(fbaddlit, $fbstr);                                             03126
      retvalue _ idplname(stid, &idstr);                                   03128
      END;                                                                 03129
   = idchr:                                                                03130
      BEGIN                                                                03131
      IF (stid _ namelook(idstid, $"'individuals'")) = endfil OR
      (stid := getsub(stid)) = stid THEN EXIT CASE;                        03132
      *fbstr* _ CR, LF, "The following individuals with last names
      beginning with the letter(s) ", *string*, " are already
      defined", CR, LF;                                                    03133
      fbctl(fbaddlit, $fbstr);                                             03134
      FOR i _ 1 UP UNTIL > string.L DO %remove leading spaces%
                                                                           03136
         IF *string*[i] # SP THEN                                          03137
            BEGIN                                                          03138
            IF i NOT= 1 THEN                                               03139
               *string* _ *string*[i TO string.L];                         03140
            EXIT;                                                          03141
            END;                                                           03142
      *tempsr* _ *string*;                                                 03143
      FOR i _ 1 UP UNTIL > tempsr.L DO %convert spaces to ''%
                                                                           03144
         IF *tempsr*[i] = SP THEN *tempsr*[i] _ '';                        03145
      astruc($tempsr); %force upper case%                                  03326
      LOOP                                                                 03146
         BEGIN                                                             03147
         IF FIND SF(stid) $SP '( $SP  '' ^tx1 ['")] ^tx2 _tx2 $SP
         "LAST NAME" THEN                                                  03148
            BEGIN                                                          03327
            *uppercasestring* _ + tx1 tx2;                                 03329
            IF FIND SF(*uppercasestring*) *tempsr* THEN                    03330
               retvalue _ retvalue + idplname(stid, &idstr);               03149
            END;                                                           03328
         IF getftl(stid)  OR inptrf := FALSE THEN EXIT LOOP;               03150
         stid _ getsuc(stid);                                              03151
         END;                                                              03152
      IF  NOT retvalue THEN                                                03153
         BEGIN                                                             03439
         *fbstr* _ CR, LF, "None", CR, LF;                                 03435
         fbctl(fbaddlit, $fbstr);                                          03436
         END;                                                              03438
      END;                                                                 03155
   = strinname: %scan names for lit%                                       03156
      BEGIN                                                                03157
      *uppercasestring* _ *string*;                                        03158
      astruc($uppercasestring);                                            03159
      %process individuals%                                                03160
         IF (stid _ namelook(idstid, $"'individuals'")) NOT=
```

```
      endfil AND (stid := getsub(stid)) NOT= stid THEN     03161
         BEGIN                                             03162
         *fbstr* _ CR, LF, "Following is a list of individuals
         which have ", *string*, " in their names:", CR, LF;
                                                           03440
         fbctl(fbaddlit, $fbstr);                          03441
         foundsome _ 0;                                    03166
         LOOP                                              03167
            BEGIN                                          03168
            IF (FIND SF(stid)  [")] $SP "LAST NAME") AND (stid
            := getsub(stid)) NOT= stid THEN                03169
               LOOP                                        03170
                  BEGIN                                    03171
                  *work* _ SF(stid) SE(stid);              03172
                  getifnf($work, $tempsr,0,0); %first name
                  first%                                   03173
                  astruc($tempsr);                         03174
                  IF FIND SF(*tempsr*) [*uppercasestring*] THEN
                                                           03175
                     BEGIN                                 03176
                     idpind($work, &idstr);                03177
                     BUMP foundsome;                       03178
                     crlf();                               03179
                     END;                                  03180
                  IF inptrf := FALSE THEN EXIT LOOP 2;     03181
                  IF getftl(stid) THEN                     03182
                     BEGIN                                 03183
                     stid _ getsuc(stid);                  03184
                     EXIT LOOP;                            03185
                     END;                                  03186
                  stid _ getsuc(stid);                     03187
                  END;                                     03188
            IF getftl(stid) OR inptrf := FALSE THEN EXIT LOOP;
                                                           03189
            stid _ getsuc(stid);                           03190
            END;                                           03191
         IF NOT foundsome THEN                             03192
            BEGIN                                          03442
            *fbstr* _ CR, LF, "None", CR, LF;              03443
            fbctl(fbaddlit, $fbstr);                       03444
            END;                                           03446
         retvalue _ retvalue + foundsome;                  03194
         END;                                              03195
      %process groups%                                     03196
         IF (stid _ namelook(idstid, $"'groups'")) NOT= endfil AND
         (stid := getsub(stid)) NOT= stid THEN             03197
            BEGIN                                          03198
            *fbstr* _ CR, LF, "Following is a list of groups which
            have ", *string*, " in their names:", CR, LF;  03447
            fbctl(fbaddlit, $fbstr);                       03448
            foundsome _ 0;                                 03202
            LOOP                                           03203
               BEGIN                                       03204
               *work* _ SF(stid) SE(stid);                 03205
               getinam($work, $tempsr,0,0);                03206
               astruc($tempsr);                            03207
```

```
                IF FIND SF(*tempsr*) [*uppercasestring*] THEN 03208
                   BEGIN                                      03209
                   idpgrporg($work, &idstr);                  03210
                   BUMP foundsome;                            03211
                   crlf();                                    03212
                   END;                                       03213
                IF getftl(stid) OR inptrf := FALSE THEN EXIT LOOP;
                                                              03214
                stid _ getsuc(stid);                          03215
                END;                                          03216
             IF NOT foundsome THEN                            03217
                BEGIN                                         03449
                *fbstr* _ CR, LF, "None", CR, LF;             03450
                fbctl(fbaddlit, $fbstr);                      03451
                END;                                          03453
             retvalue _ retvalue + foundsome;                 03219
             END;                                             03220
          %process organizations%                             03221
             IF (stid _ namelook(idstid, $"'organizations'")) NOT=
             endfil AND (stid := getsub(stid)) NOT= stid THEN 03222
                BEGIN                                         03223
                *fbstr* _ CR, LF, "Following is a list of
                organizations which have ", *string*, " in their
                names:", CR, LF;                              03454
                fbctl(fbaddlit, $fbstr);                      03455
                foundsome _ 0;                                03227
                LOOP                                          03228
                   BEGIN                                      03229
                   *work* _ SF(stid) SE(stid);                03230
                   getinam($work, $tempsr,0,0);               03231
                   astruc($tempsr);                           03232
                   IF FIND SF(*tempsr*) [*uppercasestring*] THEN 03233
                      BEGIN                                   03234
                      idpgrporg($work, &idstr);               03235
                      BUMP foundsome;                         03236
                      crlf();                                 03237
                      END;                                    03238
                   IF getftl(stid)  OR inptrf := FALSE THEN EXIT LOOP;
                                                              03239
                   stid _ getsuc(stid);                       03240
                   END;                                       03241
                IF NOT foundsome THEN                         03242
                   BEGIN                                      03456
                   *fbstr* _ CR, LF, "None", CR, LF;          03457
                   fbctl(fbaddlit, $fbstr);                   03458
                   END;                                       03460
                retvalue _ retvalue + foundsome;              03244
                END;                                          03245
          END;                                                03246
       ENDCASE;                                               03247
    rubabt _ savrubabt;                                       03248
    IF NOT fileno THEN close(idstid.stfile := 0);             03249
    RETURN(retvalue);                                         03250
    END.
                                                              03251
(idplname) %print a last name branch of individuals%
```

```
PROCEDURE (stid, idstr);                                                01746
   LOCAL count;                                                         01747
   LOCAL STRING work[2000];                                             01748
   REF idstr;                                                           01749
   count _ 0;                                                           01750
   IF (stid := getsub(stid)) = stid THEN RETURN(count);                 01751
   LOOP                                                                 01752
      BEGIN                                                             01753
      BUMP count;                                                       01754
      *work* _ SF(stid) SE(stid);                                       01755
      idpind($work, &idstr);                                            01756
      IF getftl(stid) OR inptrf := FALSE THEN EXIT LOOP;                01758
      stid _ getsuc(stid);                                              01759
      END;                                                              01760
   RETURN(count);                                                       01761
   END.                                                                 01762
                                                                        01763
(idpind) %format and print abreviated individual's ident entry%
PROCEDURE (string, idstr);                                              01764
   LOCAL STRING work[250];                                              01765
   LOCAL TEXT POINTER  orgf, orge, namef, namee;                        01766
   REF idstr;                                                           01767
   getiid(string, &idstr, 0,0);                                         01768
   getiorg(string, 0, $orgf, $orge);                                    01769
   getinam(string, 0, $namef, $namee);                                  01770
   *work* _ namef namee, ", Organization: ", orgf orge, ", Ident = ",
   *idstr*, CR, LF;                                                     01771
   fbctl(fbaddlit, $work);                                              01772
   RETURN;                                                              01773
   END.                                                                 01774
                                                                        01775
(idpgrporg) %format abbreviated group or organization entry%
PROCEDURE (string, idstr);                                              01776
   LOCAL STRING work[250];                                              01777
   LOCAL TEXT POINTER namef, namee;                                     01778
   REF idstr;                                                           01779
   getiid(string, &idstr, 0,0);                                         01780
   getinam(string, 0, $namef, $namee);                                  01781
   *work* _ namef namee, ", Ident = ", *idstr*, CR, LF;                 01782
   fbctl(fbaddlit, $work);                                              01783
   RETURN;                                                              01784
   END.                                                                 01785
                                                                        01786
(idflsearch) %search ident-file for last name or content-in-name%
PROCEDURE (type, idstr, fileno);                                        03255
   %Searches id file for individual's last name, first characters of
   individual's last names, or string, depending on type.  Returns
   true or false depending on whether an ident is accepted.. If true,
   idstr contains the ident.%                                           03256
   %------------%                                                      03257
   LOCAL  count, idstid, lnamstr, mode, term;                           03258
   LOCAL STRING newidstr[20], work[500], fbstr[1000];                   03259
   LOCAL TEXT POINTER tptr1, tptr2;                                     03260
   REF idstr;                                                           03261
   term _ 777777B;   % for calls to !rfmod and !sfmod %                 03262
   idstid _ orgstid;                                                    03263
```

```
   IF fileno THEN                                                    03264
      idstid.stfile _ fileno                                         03265
   ELSE                                                              03266
      BEGIN                                                          03267
      ON SIGNAL ELSE sigclose(idstid.stfile := 0);                   03268
      idstid.stfile _ open(0, jflname($"identfile"));                03269
      END;                                                           03270
   CASE namesearch(&idstr, $newidstr, type, idstid.stfile) OF        03271
      = 1 : %only one hit -- is it correct%                          03272
         BEGIN                                                       03273
         IF type = lname THEN                                        03274
            BEGIN                                                    03275
            *fbstr* _ CR, LF, "Is this the correct ", *idstr*, "? ",
            CR, LF;                                                  03461
            fbctl(fbaddlit, $fbstr);                                 03462
            END                                                      03279
         ELSE                                                        03280
            BEGIN                                                    03463
            *fbstr* _ CR, LF, "Is this the correct one? ";           03465
            fbctl(fbaddlit, $fbstr);                                 03466
            END;                                                     03464
         % Save the current terminal mode word to restore later; must
         call echoff to make user interaction here like other places.
         CLI must be up to something funny. %                        03281
            !rfmod(term);                                            03282
            mode _ r2;                                               03283
            echoff();                                                03284
         CASE input() OF                                             03285
            = cachar, = 'Y, = 'y:                                    03286
               BEGIN                                                 03287
               !sfmod (term, mode);                                  03288
               *idstr* _ *newidstr*;                                 03289
               *fbstr* _ "Yes",  CR, LF, "Ident ", *idstr*, "
               accepted", CR, LF;                                    03467
               fbctl(fbaddlit, $fbstr);                              03468
               IF NOT fileno THEN close(idstid.stfile := 0);         03293
               curchr _ cachar;                                      03294
               RETURN(TRUE);                                         03295
               END;                                                  03296
            ENDCASE                                                  03297
               BEGIN                                                 03298
               !sfmod (term, mode);                                  03299
               *fbstr* _ "No", CR, LF;                               03469
               fbctl(fbaddlit, $fbstr);                              03470
               IF NOT fileno THEN close(idstid.stfile := 0);         03301
               curchr _ cachar;                                      03302
               RETURN(TRUE);                                         03303
               END;                                                  03304
         END;                                                        03305
      > 1 : %more than one hit -- ask for correct one%               03306
         BEGIN                                                       03307
         *fbstr* _ CR, LF, "Type the correct IDENT: ";               03471
         fbctl(fbaddlit, $fbstr);                                    03472
         *idstr* _ NULL;                                             03309
         rdlit(&idstr, 0);                                           03310
         IF idstr.L THEN                                             03311
```

```
            BEGIN                                                   03312
            IF NOT fileno THEN close(idstid.stfile := 0);           03313
            RETURN(TRUE);                                           03314
            END;                                                    03315
         END;                                                       03316
      ENDCASE                                                       03317
         BEGIN                                                      03318
         *idstr* _ NULL;                                            03319
         END;                                                       03320
   curchr _ 'N;                                                     03321
   IF NOT fileno THEN close(idstid.stfile := 0);                    03322
   RETURN(FALSE);                                                   03323
   END.
                                                                    03324
(makgid) %generate an ident from group name string%
PROCEDURE (namestr, idstr);                                         01832
                                                                    01833
   LOCAL count;                                                     01834
   LOCAL TEXT POINTER tptr1, tptr2;                                 01835
   REF namestr, idstr;                                              01836
   count _ empty + 1;                                               01837
   *idstr* _ *namestr*[count];                                      01838
   FIND SF(*namestr*) ^tptr1 ^tptr2;                                01839
   WHILE (count _ count+1) <= idstr.M DO                            01840
      IF (FIND tptr1 [SP] [L] ^tptr1 _tptr1 ^ tptr2) THEN           01841
         *idstr* _ *idstr*, tptr1 tptr2                             01842
      ELSE EXIT LOOP;                                               01843
   RETURN;                                                          01844
   END.
                                                                    01845
(ckilmem) %verify logical membership of an ident in an ident list%
PROCEDURE (ident, l1, l2, openidfileno);                            03348
   %----------%                                                     03349
   LOCAL outcome, idfileno, nident, istid;                          03350
   LOCAL STRING grpslist [1000], grpsident [35], idstr[35],
   checkedgrps [1000];                                              03351
   LOCAL TEXT POINTER g1, g2, z1, z2, z3, z4;                       03352
   REF ident, l1, l2, lgngrps, nident;                              03353
   %----------%                                                     03354
   idfileno _ openidfileno;                                         03355
   %null list?%                                                     03356
      IF NOT (FIND l1 > (LD/'-) ^z1) OR z1 [1] > l2 [1] THEN        03357
         RETURN (FALSE);                                            03358
   %user appear explicitly in list?%                                03359
      *grpslist* _ l1 l2;                                           03360
      astruc($grpslist);                                            03361
      FIND SF(*grpslist*) ^z3 SE(*grpslist*) ^z4;                   03362
      IF ckipmem (&ident, $z3, $z4) THEN RETURN (TRUE);             03363
   %get groups from access list%                                    03364
      outcome _ FALSE;                                              03365
      &nident _ 0;                                                  03366
      g1[1] _ g2[1] _ 1; %for initial test below%                   03367
      IF NOT idfileno THEN                                          03368
         BEGIN                                                      03369
         idfileno _ open (0, jflname ($"Identfile"));               03370
         ON SIGNAL ELSE IF idfileno AND NOT openidfileno THEN       03371
```

```
         sigclose (idfileno := 0);                                    03372
      END;                                                            03373
   istid _ orgstid; istid.stfile _ idfileno;                          03374
   *grpslist* _ NULL;                                                 03375
   FIND 11 ^z2;                                                       03376
   WHILE ognxt($z1, $z2, istid) AND z2[1] <= 12[1] DO                 03377
      *grpslist* _ *grpslist*, z1 z2, " ";                            03378
   IF grpslist.L = 0 THEN GOTO ckilexit; %no groups in access
   list%                                                              03379
   astruc($grpslist);                                                 03380
   FIND SF(*grpslist*) ^z3 SE(*grpslist*) ^z4 _z4;                    03381
%locate ident's group list%                                           03382
   IF *ident* = *initsr* AND &lgngrps THEN                            03383
      FIND SF(*lgngrps*) ^g1 SE(*lgngrps*) ^g2                        03384
   ELSE                                                               03385
      &nident _ &ident;                                               03386
*checkedgrps* _ NULL;                                                 03387
WHILE g1[1] < g2[1] OR &nident DO                                     03388
   BEGIN                                                              03389
   WHILE g1[1] < g2[1] DO                                             03390
      BEGIN                                                           03391
      IF (FIND g1 > ^z1 1$( LD / '- ) ^z2 $(SP/TAB/',) ^g1) THEN
                                                                      03392
         BEGIN                                                        03393
         *idstr* _ z1 z2;                                             03394
         IF ckipmem($idstr, $z3, $z4) THEN                            03395
            BEGIN                                                     03396
            outcome _ TRUE;                                           03397
            GOTO ckilexit;                                            03398
            END;                                                      03399
         *checkedgrps* _ *checkedgrps*, z1 z2, " ";                   03400
         END                                                          03401
      ELSE FIND g1 > CH ^g1; %skip over one char--avoids infinite
      loop in a bad list%                                             03402
      END;                                                            03403
   IF &nident THEN %find super-groups of nident%                      03404
      BEGIN                                                           03405
      IF ckident (&nident:=0, 0, idfileno : istid) AND
      getigrps(istid, 0, $g1, $g2) AND g1[1] < g2[1] THEN REPEAT
      LOOP;                                                           03406
      END;                                                            03407
   IF checkedgrps.L # 0 THEN %break out last group to nident%
                                                                      03408
      BEGIN                                                           03409
      IF NOT (FIND SE(*checkedgrps*) < $NP ^z2 1$(LD/'-) ^z1) THEN
      err($"string err in ckilmem");                                  03410
      *grpsident* _ z1 z2;                                            03411
      &nident _ $grpsident;                                           03412
      checkedgrps.L _ z1[1] - 1;                                      03413
      END;                                                            03414
   END;                                                               03415
(ckilexit): IF idfileno AND NOT openidfileno THEN                     03416
   close (idfileno := 0);                                             03417
RETURN (outcome);                                                     03418
END.                                                                  03419
                                                                      03420
```

```
(ognxt) %search list of idents for next group or org.%
PROCEDURE(l1, l2, idstid);                                          03421
   %start at l1, return TRUE with text pointers l1 and l2 around
   ident on success.  return FALSE failure.  idstid must point into
   open identfile%                                                  03422
   %------------------%                                            03423
   LOCAL STRING tidstr[40];                                         03424
   REF l1, l2;                                                      03425
   LOOP                                                             03426
      BEGIN                                                         03427
      IF NOT (FIND l2 $(SP / TAB / ',) ^l1 1$( LD / '- ) ^l2 ) THEN
      RETURN(0);                                                    03428
      *tidstr* _ l1 l2;                                             03429
      IF (idstid _ namelook(idstid, $tidstr)) # endfil AND
      orgrptst(idstid, 0) THEN RETURN(TRUE);                        03430
      END;                                                          03431
   END.
                                                                    03432
(ckipmem) %verify physical membership of an ident in an ident list%
PROCEDURE (ident, l1, l2);                                          02476
   %----------%                                                    02477
   LOCAL STRING listident [35];                                     02478
   LOCAL TEXT POINTER p1, p2;                                       02479
   REF ident, l1, l2;                                               02480
   %----------%                                                    02481
   FIND l1 >;                                                       02482
   WHILE ((FIND $(SP/',) ^p1 1$(LD/'-) ^p2) AND (p2 [1] <= l2 [1]))
   DO                                                               02483
      BEGIN                                                         02484
      *listident* _ p1 p2;                                          02485
      IF *ident* = *listident* THEN RETURN (TRUE);                  02486
      END;                                                          02487
   RETURN (FALSE);                                                  02488
   END.                                                             02489
                                                                    02490
(cinidlst) %clean ident list -- of comments, expansion chars, etc.%
PROCEDURE (inlist, %appends to->% outlist);                         02491
   %----------%                                                    02492
   LOCAL STRING ident [40];                                         02493
   LOCAL TEXT POINTER l1, l2, p1, p2, p3;                           02494
   REF inlist, outlist;                                             02495
   %----------%                                                    02496
   FIND SF(*outlist*) ^l1;                                          02497
   IF outlist.L AND *outlist* [outlist.L] # SP THEN                 02498
      *outlist* _ *outlist*, SP;                                    02499
   FIND SF(*inlist*) ^p3;                                           02500
   WHILE (FIND p3 > $(SP/',) ('&/'^/) ^p1 1$(LD/'-) ^p2 ($(SP/',) '(
   [')] /) ^p3) DO                                                  02501
      BEGIN                                                         02502
      *ident* _ p1 p2;                                              02503
      FIND SE(*outlist*) ^l2;                                       02504
      IF NOT ckipmem ($ident, $l1, $l2) THEN                        02505
         *outlist* _ *outlist*, *ident*, SP;                        02506
      END;                                                          02507
   IF *outlist* [outlist.L] = SP THEN BUMP DOWN outlist.L;          02508
   RETURN;                                                          02509
```

```
    END.                                                              02510
                                                                      02511
(delcmidents) %delete idents common to both ident lists%
PROCEDURE (lhlist, lhdelete, rhlist, rhdelete);                       02512
    %----------%                                                      02513
    LOCAL STRING lhident [40], rhident [40];                          02514
    LOCAL TEXT POINTER lh1, lh2, lht, rh1, rh2, rht;                  02515
    REF lhlist, rhlist;                                               02516
    %----------%                                                      02517
    FIND SE(*lhlist*) (';/) ^lh1;                                     02518
    WHILE (FIND lh1 < ^lht $(SP/',) ^lh2 1$(LD/'-) ^lh1) DO           02519
        BEGIN                                                         02520
        *lhident* _ lh1 lh2;                                          02521
        FIND SE(*rhlist*) (';/) ^rh1;                                 02522
        WHILE (FIND rh1 < ^rht $(SP/',) ^rh2 1$(LD/'-) ^rh1) DO       02523
            BEGIN                                                     02524
            *rhident* _ rh1 rh2;                                      02525
            IF *lhident* = *rhident* THEN                             02526
                BEGIN                                                 02527
                IF lhdelete THEN ST lh1 lht _ NULL;                   02528
                IF rhdelete THEN ST rh1 rht _ NULL;                   02529
                EXIT LOOP;                                            02530
                END;                                                  02531
            END;                                                      02532
        END;                                                          02533
    IF lhdelete AND FIND SE(*lhlist*) (';/) ^lh2 1$(SP/',) ^lh1 THEN
                                                                      02534
        ST lh1 lh2 _ NULL;                                            02535
    IF rhdelete AND FIND SE(*rhlist*) (';/) ^rh2 1$(SP/',) ^rh1 THEN
                                                                      02536
        ST rh1 rh2 _ NULL;                                            02537
    RETURN;                                                           02538
    END.                                                              02539
                                                                      02540
FINISH

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

                                                                      02393
                        where all of the idents have the same lastname (the
```

```
            name of the head of the branch).                    02394
                .
                .
                .                                               02395
            .
            .
            .                                                   02396
        ('groups')                                              02397
            (ident) Expand Group
            membershiplist
            groupname
            cordinatorIdent
            Address

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

                                                                02401
            .
            .
            .                                                   02402
                                                                02403
```

INDEX LIULKP

< NLS, INDEX-L10LRP.NLS.2, >, 23-Jun-77 11:33 LLG ;;;;

```
< NLS, INDEX-XL10RUNTIME.NLS.3, >, 7-Jul-77 12:44 SKO ;;;;
((stast) RECORD stadr) <110, xl10runtime, 01422> FIELD - 18
7B4                                                              0439
((stbw) RECORD stwc) <110, xl10runtime, 01420>   FIELD - 9
7B2                                                              0437
(alostk)        <110, xl10runtime, 03541>        PROCEDURE
8J1
   FORMAL PARAMTERS( astk REF, nstksize, zone, acoroutine REF, nargs)
   % alocate & openport coroutine stack %                        0612
(apachr)        <110, xl10runtime, 0289>         LOCAL             9D
                                                                 0617
(apblnk)        <110, xl10runtime, 0705>         PROCEDURE
11N
   FORMAL PARAMTERS( ast, cnt)                                   0632
(apchr)         <110, xl10runtime, 01662>        LOCAL
11H                                                              0628
(apstore)       <110, xl10runtime, 0598>         LOCAL
11H2D                                                            0629
(aptstr)        <110, xl10runtime, 0294>         LOCAL             9E
                                                                 0618
(ascom)         <110, xl10runtime, 0608>         PROCEDURE
11J
   FORMAL PARAMTERS( astr1, astr2, relation)                     0631
(asrref)        <110, xl10runtime, 0548>         LOCAL
11A                                                              0623
(astruc)        <110, xl10runtime, 01249>        PROCEDURE
11V
   FORMAL PARAMTERS( astrng)
   %a-string to upper case%                                      0635
(bfs)           <110, xl10runtime, 0997>         LOCAL
12E                                                              0644
(bits)          <110, xl10runtime, 03920>        EXT               6E
   %                                                             0434
(bptctn)        <110, xl10runtime, 03534>        LOCAL
8E2E7
   % syscat points to catch frame being activated %              0576
(bptinv)        <110, xl10runtime, 02284>        LOCAL
8E4F3
   % syscpe points to catch frame %                              0580
(bptrcv)        <110, xl10runtime, 02294>        LOCAL
8B2
   % see arguments - label for debugging convenience %           0530
(bptres)        <110, xl10runtime, 02285>        LOCAL
8E3E8
   % (M)+4 is res1, others follow %                              0578
(bptrtn)        <110, xl10runtime, 02403>        LOCAL
8E4H8E
   % break just before real return %                             0582
(bpttrm)        <110, xl10runtime, 02259>        LOCAL
8E5F
   % here, systloc is term. location %                           0584
(bsc)           <110, xl10runtime, 0165>         LOCAL             9A
                                                                 0614
(byteptr)       <110, xl10runtime, 01520>        LOCAL             7A
                                                                 0435
(callstksize)   <110, xl10runtime, 03995>        EXT CONSTANT =2000B
```

```
8A4A                                                                    0491
(catch)         <110, xl10runtime, 02149>       CATCHPHRASE
9E21                                                                    0619
(catchnam)      <110, xl10runtime, 01606>       LOCAL
8E6A1
   % address of catchphrase %                                           0586
(catchnam)      <110, xl10runtime, 01613>       LOCAL
8E7A1
   % address of catchphrase %                                           0588
(catchname)     <110, xl10runtime, 01625>       LOCAL
8F3A1
   % address of catchphrase %                                           0593
(catchname)     <110, xl10runtime, 02263>       LOCAL
8F2A1
   % address of catchphrase %                                           0591
(catenc)        <110, xl10runtime, 02045>       CONSTANT =2
8A5C
   % enable count for catchphrase %                                     0494
(catfrm)        <110, xl10runtime, 02044>       CONSTANT =1
8A5B
   % (frame) PORT id for catchphrase %                                  0493
(catloc)        <110, xl10runtime, 02043>       CONSTANT =0
8A5A
   % location of catchphrase %                                          0492
(catmrk)        <110, xl10runtime, 03633>       CONSTANT =8
8A5I
   % indicates which signals this catchphrase has seen %               0500
(catown)        <110, xl10runtime, 02317>       CONSTANT =7
8A5H
   % (frame) PORT id for owning procedure %                             0499
(catp)          <110, xl10runtime, 03847>       REF
8F4C
   % temporary pointer for looping over catchphrase stack %            0595
(catprm)        <110, xl10runtime, 02308>       CONSTANT =6
8A5G
   % single catchphrase parameter %                                     0498
(catrtn)        <110, xl10runtime, 02047>       CONSTANT =4
8A5E
   % real return location for owning procedure %                        0496
(cats)          <110, xl10runtime, 02046>       CONSTANT =3
8A5D
   % S at invoke time %                                                 0495
(catsiz)        <110, xl10runtime, 02029>       CONSTANT =10
8A5L
   % entries in catch frame %                                           0502
(catstk)        <110, xl10runtime, 03634>       CONSTANT =9
8A5J
   % stack id of stack which owns this catchphrase %                    0501
(cattloc)       <110, xl10runtime, 02048>       CONSTANT =5
8A5F
   % termination location %                                             0497
(cct)           <110, xl10runtime, 01115>       PROCEDURE
120
   FORMAL PARAMTERS( chrcls)                                            0647
(charno)        <110, xl10runtime, 01667>       LOCAL
11G1A
```

8C2B
% used by catchphrase to contain a mask which will cause the bit corresponding to the signal which has been in progress to be shut off. %
0562
(mkbptr) <110, xl10runtime, 0863> LOCAL
11AE
0641
(mvbtbf) <110, xl10runtime, 01224> LOCAL
11S
0633
(nomap) <110, xl10runtime, 03348> LOCAL
8D1E7Q
0572
(notrce) <110, xl10runtime, 03415> LOCAL
8B6D11I
0547
(notrce1) <110, xl10runtime, 03427> LOCAL
8B6C5I
0543
(openswork) <110, xl10runtime, 0476> PROCEDURE
10B
0622
(pcall) <110, xl10runtime, 02211> LOCAL
8B6D
% port-call code (trace point) %
0544
(pclerr) <110, xl10runtime, 03650> LOCAL
8B6D12E
0553
(pcloop) <110, xl10runtime, 03612> LOCAL
8B6D12C
0550
(pclpend) <110, xl10runtime, 03625> LOCAL
8B6D12D
0552
(pdc) <110, xl10runtime, 0931> PROCEDURE
12A
FORMAL PARAMTERS(pdcnum, pntloc)
0642
(pid) <110, xl10runtime, 04183> LOCAL
8H1C
0606
(popsig) <110, xl10runtime, 02070> LOCAL
8G3
0599
(port) <110, xl10runtime, 02095> LOCAL
8H1B1
% the port id %
0605
(port) <110, xl10runtime, 02103> LOCAL
8H2B1
% the port id = frame pointer %
0609
(rdummy) <110, xl10runtime, 02366> FIELD - 1
7E6
%DEX dummy flag-- scratch space%
0458
(repet) <110, xl10runtime, 02367> FIELD - 3
7E7
%DEX repetition-- scratch space%
0459
(rhf) <110, xl10runtime, 02368> FIELD - 1
7E8
%head flag. true if this is head of plex%
0460
(ring) <110, xl10runtime, 02360> RECORD 7E
0452
(rinst1) <110, xl10runtime, 02364> FIELD - 7
7E4
%DEX interpolation string-- scratch space%
0456
(rinst2) <110, xl10runtime, 02365> FIELD - 7
7E5
%DEX interpolation string-- scratch space%
0457
(rnamef) <110, xl10runtime, 02370> FIELD - 1

```
7E10
   %name flag, true if statement has a name%                          0462
(rnameh)        <110, xl10runtime, 02373>          FIELD - 30
7E13
   %name hash for this statement%                                     0465
(rnull)         <110, xl10runtime, 02372>          FIELD - 1
7E12
   %unused%                                                           0464
(rsdb)          <110, xl10runtime, 02363>          FIELD - 18
7E3
   %psdb of sdb for this statement%                                   0455
(rsid)          <110, xl10runtime, 02374>          FIELD - 30
7E14
   %statement identifier%                                             0466
(rsub)          <110, xl10runtime, 02361>          FIELD - 18
7E1
   %psid of sub of this statment%                                     0453
(rsuc)          <110, xl10runtime, 02362>          FIELD - 18
7E2
   %psid of suc of this statement%                                    0454
(rtf)           <110, xl10runtime, 02369>          FIELD - 1
7E9
   %tail flag, true if tail of plex%                                  0461
(rtorgin)       <110, xl10runtime, 02371>          FIELD - 1
7E11
   %inferior tree origin flag, true if origin%                        0463
(schars)        <110, xl10runtime, 02380>          FIELD - 11
7F3
   %number of characters in this statement%                           0470
(sdbhead)       <110, xl10runtime, 02377>          RECORD               7F
                                                                       0467
(sgarb)         <110, xl10runtime, 02378>          FIELD - 1
7F1
   %true if this sdb is garbage%                                      0468
(sgbot)         <110, xl10runtime, 03904>          CONSTANT =8
8A71
   % "bottom" of stack for signal propogation%                        0522
(sgcat)         <110, xl10runtime, 03903>          CONSTANT =7
8A7H
   % catchphrase stack pointer for dispatched catchphrase %           0521
(sgfrm)         <110, xl10runtime, 03896>          CONSTANT =0
8A7A
   % signalling routine's fram (port id)%                             0514
(sgg2)          <110, xl10runtime, 03900>          CONSTANT =4
8A7E
   % signal parms %                                                   0518
(sgg3)          <110, xl10runtime, 03901>          CONSTANT =5
8A7F
   % signal parms %                                                   0519
(sgg4)          <110, xl10runtime, 03902>          CONSTANT =6
8A7G
   % signal parms %                                                   0520
(sggco)         <110, xl10runtime, 03897>          CONSTANT =1
8A7B
   %called catchphrase's co-return location %                         0515
(sggn)          <110, xl10runtime, 03907>          CONSTANT =11
```

```
8A7L
   % the signal number %                                              0525
(sggtp)        <110, xl10runtime, 03899>          CONSTANT =3
8A7D
   % signal type %                                                    0517
(sgnst)        <110, xl10runtime, 03906>          CONSTANT =10
8A7K
   % nested scan flag %                                               0524
(sgsig)        <110, xl10runtime, 03898>          CONSTANT =2
8A7C
   %signal value/name %                                               0516
(sgstk)        <110, xl10runtime, 03905>          CONSTANT =9
8A7J
   % stack pointer to base of stack propogating the signal %          0523
(sigrestore)   <110, xl10runtime, 03683>          PROCEDURE
8G2
   % restore global signal state from sig stack %                     0598
(sigsave)      <110, xl10runtime, 03665>          PROCEDURE
8G1
   FORMAL PARAMTERS( rloc)
   % save global signal state in signal stack %                       0597
(sigsiz)       <110, xl10runtime, 02030>          CONSTANT =12
8A7N
   % in signal in progress frame %                                    0526
(sinit)        <110, xl10runtime, 02386>          FIELD - 21
7F9
   %initials of user who created this sdb%                           0476
(sitpsid)      <110, xl10runtime, 02394>          FIELD - 18
7F12
   %PSID to head of inferior tree if any%                            0479
(slength)      <110, xl10runtime, 02379>          FIELD - 9
7F2
   %number of words in this sdb%                                     0469
(slngth)       <110, xl10runtime, 01262>          PROCEDURE
11W
   FORMAL PARAMTERS( bp1, bp2)
   %string length%                                                   0636
(slnmdl)       <110, xl10runtime, 02381>          FIELD - 7
7F4
   %left name delimiter for statement%                               0471
(sname)        <110, xl10runtime, 02384>          FIELD - 11
7F7
   %position of character after name%                                0474
(spsdb)        <110, xl10runtime, 02393>          FIELD - 18
7F11
   %PSDB of the next property data block 0=tail%                     0478
(spsid)        <110, xl10runtime, 02383>          FIELD - 18
7F6
   %psid of the statement for this sdb%                              0473
(sptype)       <110, xl10runtime, 02387>          FIELD - 15
7F10
   %property type of this data block%                                0477
(srlset)       <110, xl10runtime, 0858>           LOCAL
11AD                                                                 0640
(srm3)         <110, xl10runtime, 02504>          PROCEDURE
11Y
```

```
   FORMAL PARAMTERS( astrng REF, npad, pchar)                          0639
(srmake)        <110, xl10runtime, 0772>          LOCAL
11X                                                                    0637
(srnmdl)        <110, xl10runtime, 02382>         FIELD - 7
7F5
   %right name delimiter for statement%                                0472
(srovr)         <110, xl10runtime, 02956>         LOCAL
11X11                                                                  0638
(stackdecsize) <110, xl10runtime, 03886>          CONSTANT =10
8A6L
   % entries in stack descriptor frame at bottom of each stack %       0513
(status)        <110, xl10runtime, 01729>         LOCAL
8C1A
   % TRUE if recovering %                                              0560
(stbptr)        <110, xl10runtime, 01513>         LOCAL
111                                                                    0630
(stidr)         <110, xl10runtime, 01418>         RECORD                7B
                                                                       0436
(stime)         <110, xl10runtime, 02385>         FIELD - 36
7F8
   %date and time when this sdb created%                               0475
(stkblink)      <110, xl10runtime, 03889>         CONSTANT =-4
8A6D
   % backward link to previous stk or 0%                               0506
(stkdalo)       <110, xl10runtime, 04114>         CONSTANT =-8
8A6H
   % Stack which is deallocating this stack: 0 except in dalostk%      0510
(stklink)       <110, xl10runtime, 03888>         CONSTANT =-3
8A6C
   % forward link to next stk or 0 %                                   0505
(stkmdalo)      <110, xl10runtime, 04167>         CONSTANT =-9
8A6I
   % M for Stack which is deallocating this stack: 0 except in dalostk%
                                                                       0511
(stkmsave)      <110, xl10runtime, 03884>         CONSTANT =-1
8A6A
   % saved M ptr %                                                     0503
(stksdalo)      <110, xl10runtime, 04168>         CONSTANT =-10
8A6J
   % S for Stack which is deallocating this stack: 0 except in dalostk%
                                                                       0512
(stksig)        <110, xl10runtime, 03892>         CONSTANT =-7
8A6G
   % Number of the current signal in progress on this stack.%          0509
(stksize)       <110, xl10runtime, 03891>         CONSTANT =-6
8A6F
   % size of usable stack size of usable stack area%                   0508
(stkssave)      <110, xl10runtime, 03887>         CONSTANT =-2
8A6B
   % saved S ptr %                                                     0504
(stkzone)       <110, xl10runtime, 03890>         CONSTANT =-5
8A6E
   % allocation zone if allocated stack; 0 otherwise%                  0507
(stsdb)         <110, xl10runtime, 01421>         RECORD
7B3                                                                    0438
(stsidr)        <110, xl10runtime, 01423>         RECORD
```

```
7B5                                                                    0440
(svsybtm)       <110, xl10runtime, 04165>          LOCAL
8G3D                                                                   0601
(sysabl)        <110, xl10runtime, 01949>          LOCAL
8F1                                                                    0589
(sysbak)        <110, xl10runtime, 02100>          LOCAL
8H2                                                                    0608
(syscatch)      <110, xl10runtime, 02011>          CATCHPHRASE
8C7
   % TOP CATCHPHRASE in L10 ENVIRONMENT %                              0563
(syscent)       <110, xl10runtime, 02398>          LOCAL
8B6C
   % port entry (trace point) %                                        0541
(syscer)        <110, xl10runtime, 01590>          LOCAL
8B7C
   % coroutine called as procedure %                                   0555
(sysclr)        <110, xl10runtime, 02178>          LOCAL
8H3                                                                    0610
(syscte)        <110, xl10runtime, 01594>          LOCAL
8B7D
   % catchphrase termination error %                                   0556
(sysctn)        <110, xl10runtime, 03482>          LOCAL
8E2                                                                    0575
(sysdis)        <110, xl10runtime, 01611>          LOCAL
8E7                                                                    0587
(sysdpl)        <110, xl10runtime, 02233>          LOCAL
8F5                                                                    0596
(sysdpall)      <110, xl10runtime, 03830>          LOCAL
8F4                                                                    0594
(sysdrp)        <110, xl10runtime, 01623>          LOCAL
8F3                                                                    0592
(sysena)        <110, xl10runtime, 01604>          LOCAL
8E6                                                                    0585
(sysenc)        <110, xl10runtime, 02260>          LOCAL
8F2                                                                    0590
(sysent)        <110, xl10runtime, 02205>          LOCAL
8B6A
   % procedure entry point (trace point) %                             0538
(syshelp)       <110, xl10runtime, 01797>          LOCAL
8E1                                                                    0574
(sysnxp)        <110, xl10runtime, 01586>          LOCAL
8B7B
   % non-existant port called %                                        0554
(sysovr)        <110, xl10runtime, 01782>          LOCAL
8B6B
   % stack overflow (follows sysent) %                                 0540
(syspc1)        <110, xl10runtime, 03647>          LOCAL
8B6D9
   % pulse coroutine %                                                 0545
(syspc2)        <110, xl10runtime, 03608>          LOCAL
8B6D12
   % New port not in this stack.  Loop through all stacks %            0548
(syspc3)        <110, xl10runtime, 03658>          LOCAL
8B6D12B
   % point to first stack %                                            0549
(syspc4)        <110, xl10runtime, 04162>          LOCAL
```

```
8B6D12C5C                                                          0551
(syspop)        <110, xl10runtime, 01936>          LOCAL
8E4H
   % label to drop catchphrases on RETURN %                        0581
(sysprc)        <110, xl10runtime, 02092>          LOCAL
8H1                                                                0604
(sysrcv)        <110, xl10runtime, 01569>          LOCAL             8B
                                                                   0527
(sysrsume)      <110, xl10runtime, 01794>          LOCAL
8E3                                                                0577
(sysrtf)        <110, xl10runtime, 01572>          LOCAL
8B5C
   % return false %                                                0536
(sysrtn)        <110, xl10runtime, 01573>          LOCAL
8B5B
   % return, value in mreg %                                       0534
(sysrtt)        <110, xl10runtime, 01570>          LOCAL
8B5A
   % return true %                                                 0533
(syssys)        <110, xl10runtime, 01711>          LOCAL             8C
                                                                   0559
(syster)        <110, xl10runtime, 01733>          LOCAL
8B7E
   % catchphrase fall-thru error %                                 0557
(systi1)        <110, xl10runtime, 03319>          LOCAL
8D1E2                                                              0566
(systi2)        <110, xl10runtime, 03328>          LOCAL
8D1E3                                                              0567
(systi3)        <110, xl10runtime, 03329>          LOCAL
8D1E4                                                              0568
(systi4)        <110, xl10runtime, 03327>          LOCAL
8D1E5                                                              0569
(systi5)        <110, xl10runtime, 03330>          LOCAL
8D1E6                                                              0570
(systj3)        <110, xl10runtime, 03318>          LOCAL
8D1E1                                                              0565
(systoff)       <110, xl10runtime, 03375>          LOCAL
8D2                                                                0573
(systrc)        <110, xl10runtime, 03296>          LOCAL
8D1                                                                0564
(systrm)        <110, xl10runtime, 01628>          LOCAL
8E5                                                                0583
(systwd)        <110, xl10runtime, 03331>          LOCAL
8D1E7
   % JSR routine to write word into trace page %                   0571
(systz1)        <110, xl10runtime, 01576>          LOCAL
8B5B4                                                              0535
(systz2)        <110, xl10runtime, 01580>          LOCAL
8B5C5                                                              0537
(systz3)        <110, xl10runtime, 02208>          LOCAL
8B6A3
   % return to procedure %                                         0539
(systz4)        <110, xl10runtime, 02402>          LOCAL
8B6C4
   % return to port entry %                                        0542
(systz5)        <110, xl10runtime, 02218>          LOCAL
```

8B6D10 0546
(sysufl) <110, xl10runtime, 01780> LOCAL
8B7F
% stack underflow (strange) % 0558
(sysvoke) <110, xl10runtime, 01795> LOCAL
8E4 0579
(tcatp) <110, xl10runtime, 03795> REF
8C2A
% temporary pointer to catchframe stack % 0561
(temp) <110, xl10runtime, 03861> LOCAL
8G3C 0600
(teststk) <110, xl10runtime, 04184> LOCAL
8H1D 0607
(tsigptr) <110, xl10runtime, 03862> REF
8G3E 0602

INFTBK

```
< NLS, INPFBK.NLS.34, >, 22-Apr-78 18:01 JDH ;;;;
FILE inpfbk % L10 to <rel-nls>INPFBK % % (110,) (rel-nls,inpfbk.rel,) %
                                                                    02

%...declarations...%
                                                                    03
   REF  rawchr;                                                     04
   REF litda, msgda, tda, cflda, ltvsda, vspcda, namda, subda;      05
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, p = 7, m = 10, s = 9;   06
   DECLARE EXTERNAL                                               05003
      msmrkr = 201B,     % mouse marker caseshift code %           05014
      mslwvws = 206B,    % lowercase viewspec mouse shift code %   05015
      msupvws = 207B,    % upppercase viewspec mouse shift code %  05004
      dinchr = 0;        % saved char for marker termination %      05005
   DECLARE STRING bm = "0";                                         07
   DECLARE                                                          08
      notyet = 7, down = TRUE, up = FALSE;                          09
   DECLARE  %source codes for LSRT%                               02436
      srcnul = 0, srcstat = 1, srcstno = 2, srcsig = 3, srcast    02437
      = 4, srcdot = 5;                                            02438
   DECLARE name=1, word=2, contnt=3, spaces = 0;                    010
   DECLARE                                                          011
      igrps = (0, statem, charac, item, vector), igrpc = (0, 's, 'c, 'i,
      'v);                                                          012
   DECLARE STRING                                                   013
      statem = "Statement",                                         014
      charac = "Character",                                         015
      item = "Item",                                                016
      vector = "Vector";
                                                                    017
   EXTERNAL gtchwait;                                            010459
%...NLS Input Character Routine (Display or Typewriter)...%
                                                                    018
   (inpcuc) PROCEDURE; %input a character, force upper case%        0493
      %input a character and translate it to upper case if necessary%
                                                                    0494
      %------------------%                                         0495
      IF (curchr _ input()) IN ['a, 'z] THEN curchr _ curchr - 40B; 0496
      RETURN(curchr);                                               0497
      END.
                                                                    0498
   (input) PROCEDURE; %character input (TYPEWRITER  or DISPLAY)%   04742
      %The character is stored in curchr and returned.  QMOFF is also
      called, for DNLS, to make sure a "?" is not being displayed.%
                                                                   04743
      %----------%                                                04744
      curchr _ lookc();                                            04745
      IF (buffs _ buffs + 1) > buffsz THEN buffs _ 0;              04746
      IF nlmode = fulldisplay AND qmrkon THEN                      04747
         qmotf();                                                  04748
      RETURN(curchr);                                              04757
      END.
                                                                   04758
   (lookc) PROCEDURE; %look at a character%                        08580
      %look at next character in input buffer or tty%              08581
      %return the character without advancing the buffer pointer%  08582
      %-----------------%                                          08583
```

```
   LOCAL char;                                                          08584
   REF notetime;                                                        08585
   IF buffs = buffn % Is buffer empty? % THEN                           08586
      BEGIN % must get another char from user %                         08587
      buff[ buffn ] _ char _                                            08588
         IF nlmode = fulldisplay THEN dinptc() ELSE tinptc();           08589
      IF (buffn _ buffn + 1) > buffsz THEN buffn _ 0;                   08590
      END                                                               08591
   ELSE                                                                 08592
      char _ buff[ buffs ];                                             08593
   IF analtime THEN                                                     08594
      BEGIN                                                             08595
      !time();                                                          08596
      notetime( 101B %command start%, r1);                             08597
      analtime _ FALSE;                                                 08598
      END;                                                              08599
   RETURN(char);                                                        08600
   END.
                                                                        08601
                                                                        08602
(getchar) PROCEDURE; %get an untranslated character from user%          062
   %*****************************************************
   DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
   *****************************************************%               063
   %lowest level input routine--gets an untranslated character from
   the user.%                                                           064
   %-----------------%                                                  065
   LOCAL char;                                                          066
   rubabt _ TRUE; %allow command to be aborted%                         067
   %for ARC TENEX 131, there used to be calls to nlcrms around the
   !pbin, TRUE and FALSE respectively, E.G. %                          010460
      %IF nlcron AND nlstyp IN [0,2] THEN nlcrms(TRUE);%                068
   !pbin;                                                              069
   (gtchwait): %doctrlc can check if ^C interrupt happened here, and
   if so, diddle the interrupt return location so the "PBIN" is
   called again%                                                       010458
   IF (char_r1)=CR AND tops20flag THEN                                 010625
      BEGIN                                                            010626
      !pbin(); %skip over LF%                                          010627
      char _ EOL;                                                      010628
      END;                                                             010629
   inptrf _ inpstp _ rubabt _ FALSE;                                    072
   RETURN(char);                                                        073
   END.
                                                                        074
(lpaltgetchar) PROCEDURE; % Lineprocessor GETCHAR routine %          07544
   %*****************************************************
   DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
   *****************************************************%           07545
   % this routine used for Line Processor terminals. It calls getchar
   and filters out special codes from LP for "system reset", "printer
   string request" and "LP error as described". %                       07546
   %-----------------%                                                  07547
   LOCAL line, char, char2, lppcnt, lppfg, errecloc;                    07548
   REF ttvsda;                                                          07549
   lppfg _ lppcnt _ 0;                                                  07550
```

```
   LOOP                                                                 07551
      BEGIN                                                             07552
      (lpalt2):                                                         07553
      IF altinp.L THEN                                                  07554
         BEGIN                                                          07555
         char _ *altinp*[1];                                            07556
         *altinp* _ *altinp*[2 TO altinp.L];                            07557
         END                                                            07558
      ELSE                                                              07559
         BEGIN                                                          07560
         &rawchr _ $lpgetchar;                                          07561
         char _ getchar();                                              07562
         END;                                                           07563
      CASE char OF                                                      07564
         =176B: % special code sequence from LP %                       07565
            BEGIN                                                       07566
            (lpalt1):                                                   07567
            IF altinp.L THEN                                            07568
               BEGIN                                                    07569
               char2 _ *altinp*[1];                                     07570
               *altinp* _ *altinp*[2 TO altinp.L];                      07571
               END                                                      07572
            ELSE                                                        07573
               BEGIN                                                    07574
               &rawchr _ $lpgetchar;                                    07575
               char2 _ getchar();                                       07576
               END;                                                     07577
            CASE char2 OF                                               07578
               =177B: % user hit SYSTEM RESET %                         07579
                  BEGIN                                                 07580
                  % clear input buffer ??? %                            07581
                  %clear screen if sharing screens%                     07582
                     IF ldspjfn THEN cscreen();                         07583
                  IF (lptype .A 4M) = deltadata THEN                    07584
                     cline(0, ltvsda.datop, lpxmax);                    07585
                  !gjinf(); line _ r4;                                  07586
                  ordmbt _ curmbt _msdbit _ 0;                          07587
                  checkdevline(); %see if lp line number changed%
                                                                        010456
                  lpcmode(); % put LP in coordinate mode %              07588
                  tracksend _ 0; % initialize lp printer %              07589
                  IF lppjfn THEN % printer was open %                   07590
                     lppopen(); % open again %                          07591
                  ttywindow(ttyda); % setup TTY window %                07592
                  clrall(0, TRUE);  %assumes screen has been cleared%
                                                                        07593
                  alldsp(); % repaint whole screen %                    07594
                  dn($" ");                                             07595
                  dsubsys($ssysname);                                   07596
                  dspvsp((vspsav:=0), vspsav[1], 3);                    07597
                  char _ CD;                                            07598
                  errecloc _ getblk(16, $dspblk);                       07599
                  errfil(errecloc + bhl, 177B, 0);                      07600
                  EXIT LOOP;                                            07601
                  END;                                                  07602
               =40B: % LP requesting a string for printer %             07603
```

```
                    BEGIN                                                07604
                    IF altinp.L THEN                                     07605
                       BEGIN                                             07606
                       char _ *altinp*[1];                               07607
                       *altinp* _ *altinp*[2 TO altinp.L];               07608
                       END                                               07609
                    ELSE                                                 07610
                       BEGIN                                             07611
                       &rawchr _ $lpgetchar;                             07612
                       char _ getchar();                                 07613
                       END;                                              07614
                    IF char = 176B THEN GOTO lpalt1;                     07615
                    % should catch system restart in middle of stuff %
                                                                         07616
                    lppcnt _ lppcnt + char - 40B;                        07617
                    lppfg _ TRUE;                                        07618
                    IF SKIP !sibe(dspjfn) THEN                           07619
                       BEGIN                                             07620
                       lppsr(lppcnt);                                    07621
                       lppcnt _ 0;                                       07622
                       lppfg _ 0;                                        07623
                       END;                                              07624
                    END;                                                 07625
                 =41B:  %LP error. Next characters describe it.%         07626
                    BEGIN                                                07627
                    char _ getchar() - 40B; %count of characters in
                    error description%                                   07628
                    errecloc _ getblk(17 + char/5, $dspblk);             07629
                    errfill(errecloc + bh1, 41B, char);                  07630
                    END;                                                 07631
                 =176B: % what the... %                                  07632
                    GOTO lpalt1; % try to do right thing %               07633
                 ENDCASE                                                 07634
                    err($"undefined LP code - lpgetchar");               07635
              END;                                                       07636
           = 0: IF lppjfn THEN %printer request -- should have been a
           psi%                                                          07637
              BEGIN                                                      07638
              tracksend _ tracksend + 16;                                07639
              GOTO lpalt2;                                               07640
              END;                                                       07641
           ENDCASE EXIT LOOP;                                            07642
        END;                                                             07643
     IF lppfg THEN lppsr(lppcnt);                                        07644
     RETURN(char);                                                       07645
     END.                                                                07646
(lpgetchar) PROCEDURE; % Lineprocessor GETCHAR routine %                 07647
     %*****************************************************
     DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
     *****************************************************%             07648
     % this routine used for Line Processor terminals. It calls getchar
     and filters out special codes from LP for "system reset", "printer
     string request" and "LP error as described". %                      07649
     %-----------------%                                                 07650
     LOCAL char, lppcnt, lppfg, errecloc;                                07651
     REF ltvsda;                                                         07652
```

```
   lppfg _ lppcnt _ 0;                                                07653
   LOOP                                                               07654
      CASE (char _  getchar()) OF                                     07655
         =176B: % special code sequence from LP %                     07656
            CASE getchar() OF                                         07657
               =177B: % user hit SYSTEM RESET %                       07658
                  BEGIN                                               07659
                  % clear input buffer %                              07660
                     clrbuf(0);                                       07661
                  %clear screen if sharing screens%                   07662
                     IF ldspjfn THEN cscreen();                       07663
                  IF (lptype .A 4M) = deltadata THEN                  07664
                     cline(0, ltvsda.datop, lpxmax);                  07665
                  ordmbt _ curmbt _ msdbit _ 0;                       07673
                  checkdevline(); %see if lp line changed%           010457
                  lpcmode(); % put LP in coordinate mode %            07674
                  intter(); %send interrogate command %               07675
                  tracksend _ 0; % initialize lp printer %            07676
                  IF lppjfn THEN % printer was open %                 07677
                     lppopen(); % open again %                        07678
                  ttywindow(ttyda); % setup TTY window %              07679
                  clrall(0, TRUE);  %assumes screen has been cleared%
                                                                      07680
                  alldsp(); % repaint whole screen %                  07681
                  dn($" ");                                           07682
                  dsubsys($ssysname);                                 07683
                  dspvsp((vspsav:=0), vspsav[1], 3);                  07684
                  char _ CD;                                          07685
                  errecloc _ getblk(16, $dspblk);                     07686
                  errfill(errecloc + bhl, 177B, 0);                   07687
                  EXIT LOOP;                                          07688
                  END;                                                07689
               =40B: % LP requesting a string for printer %           07690
                  BEGIN                                               07691
                  IF (char _ getchar()) = 176B THEN REPEAT CASE;
                                                                      07692
                  % should catch system restart in middle of stuff %
                                                                      07693
                  lppcnt _ lppcnt + char - 40B;                       07694
                  lppfg _ TRUE;                                       07695
                  IF SKIP !sibe(dspjfn) THEN                          07696
                     BEGIN                                            07697
                     lppsr(lppcnt);                                   07698
                     lppcnt _ 0;                                      07699
                     lppfg _ 0;                                       07700
                     END;                                             07701
                  END;                                                07702
               =41B:  %LP error. Next characters describe it.%        07703
                  BEGIN                                               07704
                  char _ getchar() - 40B; %count of characters in
                  error description%                                  07705
                  errecloc _ getblk(17 + char/5, $dspblk);            07706
                  errfill(errecloc + bhl, 41B, char);                 07707
                  END;                                                07708
               =176B: % what the... %                                 07709
                  REPEAT CASE; % try to do right thing %              07710
```

```
            ENDCASE                                                 07711
               err($"undefined LP code - lpgetchar");               07712
         = 0: IF lppjfn THEN %printer request -- should have been a
         psi%                                                       07713
            BEGIN                                                   07714
            tracksend _ tracksend + 16;                             07715
            REPEAT CASE;                                            07716
            END;                                                    07717
         ENDCASE EXIT LOOP;                                         07718
   IF lppfg THEN lppsr(lppcnt);                                     07719
   RETURN(char);                                                    07720
   END.                                                             07721
(errfill) PROCEDURE % dump system stuff in error record %           07322
   (er, % pointer to error record %                                 07323
   errcode, % 36-bit error code %                                   07324
   dccount); %number of descriptive chars following%                07325
   LOCAL                                                            07326
      namestr[6], %contains file name for this process%             07327
      i, % for indexing %                                           07328
      sixbit, %sixbit version of a string%                          07329
      ntrys, %count attempts to put old LP record out%              07330
      dwptr, %pointer to first data word%                           07331
      lprloc, %pointer to time ring for zeroing%                   010340
      toofreq, %for frequency check of LP errors%                   07332
      pty1, % first pseudo tty %                                    07333
      ptyn, % # of them %                                           07334
      tab1, %number of tables%                                      07335
      tabn; % table number for getab %                              07336
   LOCAL STRING descrstr[200];                                      07337
   REF er, dwptr, lprngp, lprloc;                                   07338
                                                                    07339
   toofreq _ 300;                                                   07340
   IF &lprngp < $lprng0 OR &lprngp > $lprng9 THEN &lprngp _ $lprng0;
                                                                    07341
   %Try again to put out the previous record if necessary%          07342
      ntrys _ 0;                                                    07343
      IF lperrec THEN        %previous record never got out%        07344
         LOOP                                                       07345
            IF outerrec(lperrec) THEN EXIT LOOP                     07346
            ELSE                                                    07347
               BEGIN                                                07348
               ntrys _ ntrys + 1;                                   07349
               IF ntrys >= 20 THEN                                  07350
                  BEGIN                                             07351
                  dismes(1, $"Can't add LP error record to error data
                  file.  Tried 20 times.");                         07352
                  RETURN;                                           07353
                  END;                                              07354
               settimer(100, $outerrec, 0); %Wait 100 ms and try
               again.  Zero arg to outerrec causes return with no
               action -- real retry is at top of loop%             07355
               END;                                                 07356
   % store obvious stuff %                                          07358
      er.ercode _ errcode;                                          07359
      er.erlpbaud _ lpbaudfactor;                                   07360
      er.erlptype _ lptype;                                         07361
```

must have these declared

```
    % get job info %                                                    07363
      !gjinf();                                                         07364
      er.eruser _ r1;                                                   07365
      er.ercondir _ r2;                                                 07366
      er.erjobn _ r3;                                                   07367
      er.erline _ r4;                                                   07368
    % get time & date %                                                 07370
      er.erdatime _ gtadcall();                                         07372
    % get this host number %                                            07374
      sixbit _ getsbn($"LHOSTN");                                       07375
      !sysgt(sixbit);                                                   07376
      er.erhost _ r1;                                                   07377
      tabn _ r2.RH;                                                     07378
      IF NOT SKIP !getab(1B6+tabn) THEN r1 _ 0;                         07379
      pty1 _ r1.RH;                                                     07380
      ptyn _ -r1.LH;                                                    07381
      ptyn.LH _ 0;                                                      07382
    % get remote host or TIP address %                                  07384
      er.ertipn _ 0;                                                    07385
      IF er.erline IN [pty1,pty1+ptyn) THEN                             07386
         BEGIN                                                          07387
         (ef51):                                                        07388
         sixbit _ getsbn($"NETBUF");                                    07389
         !sysgt(sixbit);                                                07390
         tabn _ r2.RH;                                                  07391
         tab1 _ -r2.LH;                                                 07392
         tab1.LH _ 0;                                                   07393
         FOR i_0 UP UNTIL >= tab1 DO                                    07394
            BEGIN                                                       07395
            (ef52):                                                     07396
            IF NOT SKIP !getab(i*1B6+tabn) THEN EXIT;                   07397
            IF er.erline = r1 THEN                                      07398
               BEGIN                                                    07399
               (ef53):                                                  07400
               sixbit _ getsbn($"NETAWD");                              07401
               !sysgt(sixbit);                                          07402
               tabn _ r2.RH;                                            07403
               IF SKIP !getab(i*1B6+tabn) THEN er.ertipn _ r1.oc2qtr;
                                                                        07404
               EXIT LOOP;                                               07405
               END;                                                     07406
            END;                                                        07407
         END;                                                           07408
    % get the system name %                                             07410
      !rmap(4B11+ ($ldchr/512) );                                       07411
      i _ r1.LH; % jfn %                                                07412
      !jfns($namestr+1 .V 18M6, i, 0);                                  07413
      % updated str ptr in 1 %                                         07414
      ^r1 _ 0;                                                          07415
      er.erfns1 _ namestr[1];                                           07416
      er.erfns2 _ namestr[2];                                           07417
      er.erfns3 _ namestr[3];                                           07418
      er.erfns4 _ namestr[4];                                           07419
      er.erfns5 _ namestr[5];                                           07420
      er.erfns6 _ namestr[6];                                           07421
    % get the extra error description if any %                         07423
```

```
      er.erdlen _ dccount;                                          07424
      &dwptr _ &er + 16;              %points to first data word%
                                                                    07425
      descrstr.L _0;                                                07426
      IF dccount > 0 THEN                                           07427
         BEGIN                                                      07428
         FOR i _ 1 UP UNTIL > dccount DO                            07429
            *descrstr* _ *descrstr*, getchar() - 40B;               07430
         FOR i _ 1 UP UNTIL > (dccount + 4)/5 DO                    07431
            dwptr[i-1] _ descrstr[i];                               07432
         END;                                                       07433
   % update the lp error ring and check for too frequent errors %
                                                                    07435
      IF er.erdatime - toofreq < lprngp THEN                        07436
         BEGIN                                                     010341
         dismes(1, $"Ten Line Processor errors have occurred in the
         last five minutes.");                                     010342
         FOR &lprloc _ $lprng0 UP 1 UNTIL > $lprng9 DO             010343
            lprloc _ 0;                                            010344
         END;                                                      010345
      lprngp _ er.erdatime;                                         07438
      IF &lprngp = $lprng9 THEN &lprngp _ $lprng0                   07439
         ELSE &lprngp _ &lprngp + 1;                                07440
   % try to write out the completed record %                        07442
      IF NOT outerrec(&er) THEN                                     07443
         settimer(100, $outerrec, &er);  %Wait 100 ms and try again.%
                                                                    07444
   RETURN;                                                          07445
   END.                                                             07446
(outerrec) PROCEDURE %write the completed record in a system file%
                                                                    07503
   (er); %pointer to error record %                                 07504
   LOCAL lpjfn, erbs;                                               07505
   LOCAL STRING error[200];                                         07506
   REF er;                                                          07507
                                                                    07508
   lpjfn _ 0;                                                       07509
   lperrec _ &er;    %Not put out successfully%                     07510
   IF &er = 0 THEN RETURN (TRUE);                                   07511
   IF NOT lpjfn _ sgtjfn (0, $"<NETSYS>LPERRORS.DATA;P777724",
   $error) THEN                                                     07512
      BEGIN                                                         07513
      *error* _ "Can't get JFN for LP error data file:  ", *error*;
                                                                    07514
      dismes (1,$error);                                            07515
      lperrec _ 0;                                                  07516
      RETURN(TRUE);                                                 07517
      END;                                                          07518
                                                                    07519
   IF NOT sysopen (lpjfn, append, bintyp, $error) THEN              07520
      BEGIN                                                         07521
      IF NOT SKIP !rljfn(lpjfn) THEN NULL;                          07522
      IF FIND SF(*error*) ["is busy"] THEN                          07523
         BEGIN                          %Busy. Retry later.%       07524
         RETURN(FALSE);                                             07525
         END                                                        07526
```

```
      ELSE                                                          07527
         BEGIN                        %Real error.  Abort.%         07528
         *error* _ "Trying to record LP error:  ", *error*;         07529
         dismes(1,$error);                                          07530
         lperrec _ 0;                                               07531
         RETURN(TRUE);                                              07532
         END;                                                       07533
      END;                                                          07534
                                                                    07535
   erbs _ &er - 1;                                                  07536
   erbs.LH _ 004400B;                                               07537
   !sout(lpjfn, erbs, -(16 + (er.erdlen + 4)/5));    %Put the
   "string" out%                                                    07538
   lperrec _ 0;                                                     07539
   freeblk(&er - bhl, $dspblk);                                     07540
   IF NOT sysclose(lpjfn, $error) THEN dismes(1, $error);           07541
   RETURN(TRUE);                                                    07542
   END.                                                             07543
%...Display NLS Input Character Routines...%
                                                                     0175
  (dinptc) PROCEDURE; %DNLS single character input routine%         07750
   %get and return a character from the user, taking care of special
   input (viewspecs and markers).  It does not return until a real
   character for the main part of NLS has been read.%               07751
   %------------------%                                             07752
   LOCAL char, da, tflag;                                           07753
   LOCAL STRING inmkr[50];                                          07754
   REF da;                                                          07755
   REF  rawchr;                                                     07756
   % deleted 2 pages of tenex<13200 code here; 15-OCT-75  KJM %
                                                                   010347
   bugreg _ endfil;                                                 07856
   CASE char _ ( IF dinchr THEN dinchr := FALSE ELSE rawchr() ) OF
                                                                    07857
      # lpresc:                                                     07858
         CASE curmbt OF                                             07859
            = 0: RETURN( trnsli[char] );                            07860
            = 1: % right button down %                              07901
               BEGIN                                                07902
               RETURN( trnsli[char] );                              07903
               END;                                                 07906
            = 2: % center button down - case shift 1 %              07884
               BEGIN                                                07885
               msdbit _ TRUE;                                       07886
               CASE char OF                                         07887
                  IN ['a, 'z]:  char _ char .A 137B;                07888
                  = ',: char _ '<;                                  07889
                  = '.: char _ '>;                                  07890
                  = ';: char _ ':;                                  07891
                  = '?: char _ '\;                                  07892
                  = SP: char _ TAB;                                 07893
                  ENDCASE                                           07894
                     BEGIN                                          07895
                     dismes( 1000, $"undefined input");             07896
                     REPEAT CASE 3;                                 07897
                     END;                                           07898
```

```
      RETURN( trnsli[char] );                                    07899
      END;                                                       07900
   = 3: % right & center buttons down %                          07924
      BEGIN                                                      07925
      msdbit _ TRUE;                                             07926
      dismes( 1000, $"undefined input");                         07927
      REPEAT CASE 2;                                             07928
      END;                                                       07929
   = 4: % left button down - case shift 2 %                      07907
      BEGIN                                                      07908
      msdbit _ TRUE;                                             07909
      CASE char OF                                               07910
         IN ['a, 'z]:  char _ cssh2[ char - 141B ];              07911
         = ',: char _ '[;                                        07912
         = '.: char _ '];                                        07913
         = ';: char _ '_;                                        07914
         = '?: char _ ALT;                                       07915
         = SP: char _ CR;                                        07916
         ENDCASE                                                 07917
            BEGIN                                                07918
            dismes( 1000, $"undefined input");                   07919
            REPEAT CASE 3;                                       07920
            END;                                                 07921
      RETURN( trnsli[char] );                                    07922
      END;                                                       07923
   = 5: % right & left buttons down - markers %                  07861
      BEGIN                                                      07862
      msdbit _ TRUE;                                             07863
      %set up for marker collection%                             07864
         *inmkr* _ trnsli[char];                                 07865
      LOOP                                                       07866
         BEGIN                                                   07867
         CASE char _ rawchr() OF                                 07868
            =CD: % get out of marker collection %                07869
               RETURN(CD);                                       07870
            # lpresc: %normal character%                         07871
               *inmkr* _ *inmkr*, trnsli[char];                  07872
            ENDCASE                                              07873
               BEGIN                                             07874
               dinchr _ char;                                    07875
               EXIT LOOP;                                        07876
               END;                                              07877
         END;                                                    07878
      IF inmkr.L NOT= empty THEN %get psid, character count
      from file marker table%                                    07879
         IF (bugreg _ lkmkr($inmkr, lcfile() : bugreg[1])) =
         endfil THEN                                             07880
            err($"No such marker");                              07881
      RETURN(CA);                                                07882
      END;                                                       07883
   = 6, % left & center buttons down - lower viewspecs %
                                                                 07930
   = 7: % all buttons down - upper case viewspecs %              07931
      BEGIN                                                      07932
      msdbit _ TRUE;                                             07933
      % for upper case viewspecs %                               07934
```

```
        tflag _ IF (curmbt = 6) THEN TRUE ELSE FALSE;
                                                                    07935
    %set up for viewspecs%                                          07936
      IF NOT &da _ lda() THEN                                       07937
        err($"No such Display Area");                               07938
      dspvsp(sysvspec_da.davspec, sysvspec[1]_da.davspc2,
      3);                                                           07939
        %display the viewspecs to be affected%                      07940
      ltl();  %large viewspecs%                                     07941
      RESET savevspec;                                              07942
      *vspstr* _ NULL;                                              07943
    LOOP %read the viewspec characters%                             07944
      BEGIN                                                         07945
      CASE char OF                                                  07946
        =CD: % get out of viewspec mode %                           07947
          BEGIN                                                     07948
          dendvw();                                                 07949
          RETURN(CD);                                               07950
          END;                                                      07951
        # lpresc: %normal character%                                07952
          BEGIN                                                     07953
          char _ trnsli[char];                                      07954
          % for upper case viewspecs %                              07955
              IF (NOT tflag) AND (char IN ['a,'z]) THEN
                                                                    07956
                char _ char .A 137B;                                07957
          ON SIGNAL ELSE                                            07958
            BEGIN                                                   07959
            ON SIGNAL ELSE;                                         07960
            dendvw();                                               07961
            END;                                                    07962
          inpvsp(char);                                             07963
          ON SIGNAL ELSE;                                           07964
          IF inptrf THEN                                            07965
            BEGIN                                                   07966
            dendvw();                                               07967
            REPEAT CASE 3;                                          07968
            END;                                                    07969
          END;                                                      07970
        ENDCASE                                                     07971
          BEGIN                                                     07972
          char _ dinbc();                                           07973
          CASE curmbt OF                                            07974
            = 6, = 7:                                               07975
              BEGIN                                                 07976
              tflag _                                               07977
                IF (curmbt = 6) THEN TRUE ELSE
                FALSE;                                              07978
              IF char THEN REPEAT LOOP;                             07979
              END;                                                  07980
            ENDCASE                                                 07981
              BEGIN                                                 07982
              dendvw();                                             07983
              IF char THEN REPEAT CASE 4 (char)                     07984
              ELSE REPEAT CASE 4;                                   07985
              END;                                                  07986
```

```
                        END;                                              07987
                     char _ rawchr();                                     07988
                     END;                                                 07989
                  END;                                                    07990
               ENDCASE;                                                   07991
         ENDCASE                                                          07992
            IF char _ dinbc() THEN                                        07993
               BEGIN                                                      07994
               CASE char OF                                               07995
                  = lpresc: RETURN( trnslit[char] );                      07996
                  ENDCASE REPEAT CASE 2 (char);                           07997
               END                                                        07998
            ELSE REPEAT CASE;                                             07999
   END.

                                                                          08001
(dinbc) PROCEDURE; % read a big character %                              010794
   LOCAL char, code, seqn, dvcod, count;                                  010795
   REF rawchr, lppch;                                                     010796
   CASE code _ rawchr() OF                                                010797
      NOT IN [41B, 50B]:                                                  010798
         BEGIN                                                            010799
         !esout(18M6 .V (1 + $"Transmission Error - Second character
         lost in a big character"));                                      010800
         (resynch):                                                       010801
         UNTIL (rawchr() = lpresc) DO NULL;                               010802
         REPEAT CASE;                                                     010803
         END;                                                             010804
      ENDCASE;                                                            010805
   CASE code OF                                                           010806
      = fcor2:                                                            010807
         BEGIN                                                            010808
         gcords.xcord _ getcoord();                                       010809
         gcords.ycord _ getcoord();                                       010810
         RETURN( FALSE );                                                 010811
         END;                                                             010812
      = fcor4:                                                            010813
         BEGIN                                                            010814
         gcords.xcord.xc1 _ rawchr() - 40B;                               010928
         gcords.xcord.xc2 _ rawchr() - 40B;                               010929
         gcords.ycord.yc1 _ rawchr() - 40B;                               010930
         gcords.ycord.yc2 _ rawchr() - 40B;                               010931
         RETURN( FALSE );                                                 010819
         END;                                                             010820
      = acor2:                                                            010821
         BEGIN                                                            010822
         char _ rawchr() - 140B;                                          010823
         CASE char OF                                                     010824
            IN [0, 37B], IN [-40B, -31B]:                                 010825
               NULL;                                                      010826
            ENDCASE                                                       010827
               BEGIN                                                      010828
               !esout(18M6 .V (1 + $"Transmission Error - Third
               character lost in big character"));                        010829
               GOTO resynch;                                              010830
               END;                                                       010831
         IF char >= 0 THEN                                                010832
```

```
        BEGIN  % control char + coors %                                  010833
        gcords.xcord _ getcoord();                                       010834
        gcords.ycord _ getcoord();                                       010835
        RETURN( char );                                                  010836
        END;                                                             010837
     ordmbt _ (curmbt _ char + 40B) .V ordmbt;                           010838
     IF curmbt # 0 THEN                                                  010839
        BEGIN                                                            010840
        gcords.xcord _ getcoord();                                       010841
        gcords.ycord _ getcoord();                                       010842
        END                                                              010843
     ELSE                                                                010844
        BEGIN                                                            010845
        getcoord();                                                      010846
        getcoord();                                                      010847
        END;                                                             010848
     IF curmbt THEN RETURN( FALSE );                                     010849
     char _ msetblCordmbt := 0];                                         010850
     IF (msdbit := FALSE) THEN RETURN( FALSE )                           010851
     ELSE RETURN( char );                                                010852
     END;                                                                010853
  = acor4:                                                               010854
     BEGIN                                                               010855
     char _ rawchr() - 140B;                                             010856
     CASE char OF                                                        010857
        IN [0, 37B], IN [-40B, -31B]:                                    010858
           NULL;                                                         010859
        ENDCASE                                                          010860
           BEGIN                                                         010861
           !esout(18M6 .V (1 + $"Transmission Error - Third
           character lost in big character"));                           010862
           GOTO resynch;                                                 010863
           END;                                                          010864
     IF char >= 0 THEN                                                   010865
        BEGIN                                                            010866
        gcords.xcord.xc1 _ rawchr() - 40B;                               010867
        gcords.xcord.xc2 _ rawchr() - 40B;                               010868
        gcords.ycord.yc1 _ rawchr() - 40B;                               010869
        gcords.ycord.yc2 _ rawchr() - 40B;                               010870
        RETURN( char );                                                  010871
        END;                                                             010872
     ordmbt _ (curmbt _ char + 40B) .V ordmbt;                           010873
     IF curmbt # 0 THEN                                                  010874
        BEGIN                                                            010875
        gcords.xcord.xc1 _ rawchr() - 40B;                               010876
        gcords.xcord.xc2 _ rawchr() - 40B;                               010877
        gcords.ycord.yc1 _ rawchr() - 40B;                               010878
        gcords.ycord.yc2 _ rawchr() - 40B;                               010879
        END                                                              010880
     ELSE                                                                010881
        BEGIN                                                            010882
        rawchr();                                                        010883
        rawchr();                                                        010884
        rawchr();                                                        010885
        rawchr();                                                        010886
        END;                                                             010887
```

```
          IF curmbt THEN RETURN( FALSE );                          010888
          char _ msetbl(ordmbt := 0);                              010889
          IF (msdbit := FALSE) THEN RETURN( FALSE )                010890
          ELSE RETURN( char );                                     010891
          END;                                                     010892
       = cnch2:  %control chars on IMLACS%                         010893
          RETURN ( rawchr() - 140B );                              010894
       = irep:                                                     010895
          BEGIN                                                    010896
          lpxmax _ getcoord();                                     010897
          lpymax _ getcoord();                                     010898
          lptype _ getcoord();                                     010899
          lpdlpad _ getcoord();                                    010900
          lpbaudfactor _ getcoord();                               010901
          RETURN( FALSE )                                          010902
          END;                                                     010903
       = 47B: %new lpprint request%                                010904
          BEGIN                                                    010905
          dvcod _ rawchr();                                        010906
          seqn _ rawchr();                                         010907
          count _ rawchr() - 40B;                                  010908
          lppch(seqn, count, dvcod);                               010909
          END;                                                     010910
       = 50B: %new lp reset protocal%                              010911
          BEGIN                                                    010912
          lpsysreset();                                            010913
          RETURN(CD);                                              010914
          END;                                                     010915
       ENDCASE;                                                    010916
    RETURN( FALSE );                                               010917
    END.

                                                                   010918
(getcoord) PROCEDURE;                                              010919
    LOCAL char;                                                    010920
    IF (char_rawchr())#cooresc THEN RETURN(char-40B);              010921
    char _ 0;                                                      010922
    char.xc1 _ rawchr() - 40B;                                     010923
    char.xc2 _ rawchr() - 40B;                                     010924
    RETURN(char);                                                  010925
    END.                                                           010926
(lpsysreset)PROCEDURE; %cleanup after lp system reset%             010771
    LOCAL errecloc;                                                010772
    % clear input buffer ??? %                                     010773
    %clear screen if sharing screens%                              010774
       IF ldspjfn THEN cscreen();                                  010775
    IF (lptype .A 4M) = deltadata THEN                             010776
       cline(0, ltvsda.datop, lpxmax);                             010777
    ordmbt _ curmbt _msdbit _ 0;                                   010778
    checkdevline(); %see if lp line number changed%                010779
    lpcmode(); % put LP in coordinate mode %                       010780
    tracksend _ 0; % initialize lp printer %                       010781
    IF lppjfn THEN % printer was open %                            010782
       lppnopen(TRUE); % open again %                              010783
    ttywindow(ttyda); % setup TTY window %                         010784
    clrall(0, TRUE);  %assumes screen has been cleared%            010785
    alldsp(); % repaint whole screen %                             010786
```

```
      dn($" ");                                                    010787
      dsubsys($ssysname);                                          010788
      dspvsp((vspsav:=0), vspsav[1], 3);                           010789
      errecloc _ getblk(16, $dspblk);                              010790
      errfil(errecloc + bhl, 177B, 0);                             010791
      RETURN;                                                      010792
      END.                                                         010793
   (dendvw) PROCEDURE;  % handle end of viewspec input %            04990
      LOCAL da;                                                    04991
      REF da;                                                      04992
      IF NOT &da _ lda() THEN                                      04993
         err($"No such Display Area");                             04994
      da.davspec _ cspvs _ sysvspec;                               04995
      da.davspc2 _ cspvs[1] _ sysvspec[1];                         04996
      lts();  %small viewspecs%                                    04997
      dn($"");                                                     04998
      RESET savevspec;                                             04999
      *vspstr* _ NULL;                                             05000
      RETURN;                                                      05001
      END.
                                                                   05002
%...Typewriter NLS input routines...%                              01413
   (tinptc) PROCEDURE; %return a translated, shifted char%         01904
      %-----------------%                                          01905
      LOCAL                                                        01906
         char;   %current translated character%                    01907
      CASE char _ trnsli[rawchr()] OF %translated char%            01908
         = nullch: REPEAT CASE;                                    01909
         = cshift: %force next character upper case%               01910
            BEGIN                                                  01911
            char _ trnsli[rawchr()];                               01912
            IF char IN ['a, 'z] THEN char _ char - 40B;            01913
            RETURN(char);                                          01914
            END;                                                   01915
         = wshift: %force next word upper case%                    01916
            BEGIN                                                  01917
            IF (char _ trnsli[rawchr()]) = wshift THEN RETURN(char);
                                                                   01918
            wordshift _ TRUE;                                      01919
            REPEAT CASE(char);                                     01920
            END;                                                   01921
         IN ['a, 'z]: %normal lower alpha character%               01922
            RETURN(IF wordshift THEN char - 40B ELSE char);        01923
         IN ['A, 'Z], IN ['0, '9]: %normal upper alpha or numeric
         character%                                                01924
            RETURN(char);                                          01925
         IN [SP, '_]: %normal punctuation or space character%      01926
            BEGIN                                                  01927
            wordshift _ FALSE;                                     01928
            RETURN(char);                                          01929
            END;                                                   01930
         = BC:  NULL;                                              01931
         = BW:  wordshift _ FALSE;                                 01932
         = CA:  wordshift _ FALSE;                                 01933
         = CD:  wordshift _ FALSE;                                 01934
         = $ascbst: wordshift _ FALSE;                             01935
```

```
      IN [TAB, CR], =EOL:                                                01936
         BEGIN                                                           01937
         %  IF NOT echofg THEN todco(char); echo it%  %COMMENTED OUT
         BY CHI%                                                         01938
         wordshift _ FALSE;                                              01939
         RETURN(char);                                                   01940
         END;                                                            01941
      = C.: wordshift _ FALSE;                                           01942
      ENDCASE wordshift _ FALSE;                                         01943
   IF echofg THEN typectl(char); %echo it%                               01944
   RETURN(char); %special character%                                     01945
   END.
                                                                         01946
(txtlit) PROCEDURE(astrng); %passed the address of a string, appends
text from keyboard input buffer to string%                               01950
   % This procedure reads a literal into the string whose address is
   passed.  It also checks for control-y (enter alt mode) and returns
   alterv as the completion code. %                                      01951
   %-----------------%                                                   01952
   LOCAL rcndtn;                                                         01953
   echolt();                                                             01954
    IF inprompt THEN typeas($"T: ");                                     01955
   rcndtn _ rdlit(astrng, 0);                                            01956
   echoff(); % turn break back on, echo off%                             01957
   RETURN (rcndtn);                                                      01958
   END.                                                                  01959
                                                                         01960
(rdlit) PROCEDURE (astrng, chrlst);  %read literal from keyboard%
                                                                         09662
   %read a literal from the keyboard, doing all of the control
   things, plus breaking on any control character identified by the
   string &chrlst%                                                       09663
   %types message if buffer overflows,                                   09664
   returns                                                               09665
      1 if normal termination (ca or c.),                                09666
      2 if a break character was input,                                  09667
      3 if a BC or BW was input and the string was already empty
                                                                         09668
   GLOBALS used for DNLS literal collection:                             09669
      litstore, litstring, tabstore, littabs, litline, litcolumn,
      spacecol                                                           09670
   %                                                                     09671
   %-----------------%                                                   09672
   LOCAL bccari, col, char, nextchar, i, spclcharstr, breaktbl,
   display, nmfnd, exfnd, jfn, ind, bytptr, remchar, svcfo, sveso,
   rtjtbd, locbreaktbl[128];                                             09673
   LOCAL TEXT POINTER tp1, tp2, dp1, dp2,altd1,altd2,alte1,alte2;
                                                                         09674
   LOCAL STRING bkdupchrs[150], filstr[200], dirstr[200],extstr[10];
                                                                         09675
   REF tda, chrlst, astrng, spclcharstr, breaktbl;                       09676
   litstring _ &astrng;                                                  09692
   *extstr* _ *nlsext*;                                                  09693
   FIND SF(*altdir*) ^altd1;                                             09694
   FIND SF(*altext*) ^alte1;                                             09695
   display _ IF nlmode = fulldisplay THEN TRUE ELSE FALSE;               09696
```

```
   rtjtbd _ FALSE;  % reset right justified tab mode %                    09697
   %assumes monitor echoing for TNLS%                                     09698
      IF NOT display AND clpsw THEN echoff();                             09699
   litstore _ chbptr(astrng.L) + &astrng;                                 09700
      %for fast character appends and word-boundry line breaks%           09701
   IF NOT display AND NOT clpsw AND spftab % AND Literal TYPEIN for a
   structure manipulation command % THEN                                  09702
      BEGIN                                                               09703
      %IF feedback levadjust is on THEN <print new statement number>
      %                                                                   09704
      crlf();                                                             09705
      END;                                                                09706
   IF astrng.L AND NOT clpsw THEN                                         09707
      IF display THEN aplit(&astrng)                                      09708
      ELSE echo(&astrng); %echo knows about halfduplex%                   09709
   IF NOT clpsw AND NOT display AND buffs NOT= buffn THEN typebuff();
                                                                          09710
      %typebuff knows about halfduplex%                                   09711
   IF (&chrlst AND chrlst.L) OR slink OR spftab THEN  %must add user
   break characters to system break charaters%                            09712
      BEGIN                                                               09713
      mvbtbf( $sysbreaktbl, $locbreaktbl, 128);                           09714
         %block transfer system table to local spcae%                     09715
      IF &chrlst AND chrlst.L THEN                                        09716
         BEGIN                                                            09717
         CCPOS SF(*chrlst*);                                              09718
         FOR i _ chrlst.L DOWN UNTIL <= 0 DO                              09719
            BEGIN                                                         09720
            char _ READC;                                                 09721
            locbreaktbl[char] _ $userbreak;                               09722
            END;                                                          09723
         END;                                                             09724
      IF slink THEN                                                       09725
         BEGIN                                                            09726
         locbreaktbl['<ESC>] _ $rdlesc;                                   09727
         IF NOT display THEN                                              09728
            BEGIN                                                         09729
            svcfo _ trnslo['<^F>] := nullch;                              09730
            sveso _ trnslo['<ESC>] := nullch;                             09731
            END;                                                          09732
         END;                                                             09733
      IF spftab THEN locbreaktbl[TAB] _ $rdlspftb;                        09734
      &breaktbl _ $locbreaktbl;                                           09735
      END                                                                 09736
   ELSE &breaktbl _ $sysbreaktbl;                                         09737
   LOOP  %process a character at a time%                                  09738
      BEGIN                                                               09739
      IF nldevice = devlproc AND NOT tracking AND SKIP !sibe(100B)
      THEN                                                                09740
         track();                                                         09741
      char _ input();                                                     09742
      GOTO [breaktbl[char]];                                              09743
      (rdladdchar): %normal character%                                    09744
         IF astrng.L >= astrng.M THEN                                     09745
            BEGIN                                                         09746
            dismes(2, $"literal too long -- CDOT simulated");             09747
```

```
            curchr _ char _ C.;                                          09748
            GOTO rdlcacdot;                                              09749
            END;                                                         09750
         IF rtjtbd THEN  % right justify to tab stop %                   09751
            BEGIN                                                        09752
            <sprjtb>(char,&astrng,$bkdupchrs,col,$rtjtbd);               09753
               %rtjtbd may be cleared by sprjtb%                        010936
            litstore _ <chbptr>(astrng.L) + &astrng;                     09754
            END                                                          09755
         ELSE                                                            09756
            BEGIN                                                        09757
            %*astrng* _ *astrng*, char;%                                 09758
               ^litstore _ char;                                         09759
               BUMP astrng.L;                                            09760
            IF display AND NOT clpsw THEN litchr(char); %echo char in
            literal area%                                                09761
            END;                                                         09762
         REPEAT LOOP;                                                    09763
      (rdlspclchar): %cntrl char after CNTRL-V%                         010348
         IF char = lpresc AND NOT (char _ dinbc()) THEN  % get a big
         char %                                                          010349
            BEGIN %big char is null%                                     010932
            char _ rawchr();                                             010934
            GOTO rdlspclchar;                                            010935
            END;                                                         010933
         IF astrng.L >= astrng.M THEN                                    010357
            BEGIN                                                        010358
            dismes(2, $"literal too long -- CDOT simulated");            010359
            curchr _ char _ C.;                                          010360
            GOTO rdlcacdot;                                              010361
            END;                                                         010362
         %*astrng* _ *astrng*, char;%                                    010363
            ^litstore _ char;                                            010364
            BUMP astrng.L;                                               010365
         IF NOT clpsw THEN litchr(char); %echo char in literal area%
                                                                         010366
         REPEAT LOOP;                                                    010367
      (rdlbc):      %backspace character%                               09764
         IF astrng.L = empty THEN RETURN(3, char) ;                      09765
         char _ .litstore; %last char input%                             09766
         IF rtjtbd AND bkdupchrs.L > 0 THEN                              09767
            BEGIN                                                        09768
            astrng.L _ astrng.L - bkdupchrs.L;                           09769
            BUMP DOWN bkdupchrs.L;                                       09770
            *astrng* _ *astrng*, SP, *bkdupchrs*;                        09771
            IF display AND NOT clpsw THEN  %back up literal area and
            re-type %                                                    09772
               BEGIN                                                     09773
               FOR i_0 UP UNTIL > bkdupchrs.L DO <litbc>(char, SP);
                                                                         09774
               <litchr>(SP);                                             09775
               CCPOS SF(*bkdupchrs*);                                    09776
               UNTIL (char_READC) = ENDCHR DC <litchr>(char);            09777
               END;                                                      09778
            END                                                          09779
         ELSE                                                            09780
```

```
        BEGIN                                                   09781
        rtjtbd _ FALSE;                                         09782
        BUMP DOWN astrng.L;                                     09783
        %back up litstore%                                      09784
           r1 _ litstore;                                       09785
           r2 _ litstore-1;                                     09786
           UNTIL r2 = r1 DO                                     09787
              BEGIN                                             09788
              r3 _ r2;                                          09789
              !IBP r2;                                          09790
              END;                                              09791
           litstore _ r3;                                       09792
        IF display AND NOT clpsw THEN  %back up literal area%
                                                                09793
           litbc(char, .litstore);                              09794
        END;                                                    09795
     IF NOT display AND NOT clpsw THEN                          09796
        IF rtjtbd THEN                                          09797
           IF bkdupchrs.L > 0 THEN !pbout(SP) % we are inserting
           right justified text and "back up" means to go forward
           %                                                    09798
           ELSE                                                 09799
              BEGIN                                             09800
              !pbout(8);                                        09801
              echon();                                          09802
              END                                               09803
        ELSE IF spftab AND formln(&astrng) THEN % "back up"
        carriage %                                              09804
           BEGIN                                                09805
           !pbout(8); %back up over character echoed in response
           to BC %                                              09806
           <sptbbc>(char, &astrng); % back up over previous
           character %                                          09807
           END                                                  09808
        ELSE IF NOT rtjtbd THEN %echo what is being backed over%
                                                                09809
           BEGIN                                                09810
           CASE char OF                                         09811
              IN [SP, "z]: %normal character%                   09812
                 !pbout(char);                                  09813
              ENDCASE %Special non-printing character%          09814
                 BEGIN                                          09815
                 &spclcharstr _ npstrad(char);                  09816
                 typeas(&spclcharstr);                          09817
                 END;                                           09818
           END;                                                 09819
     REPEAT LOOP;                                               09820
  (rdlbw):      %backspace word%                                09821
     IF astrng.L = empty THEN RETURN(3, char) ;                 09822
     IF rtjtbd THEN                                             09823
        BEGIN                                                   09824
        IF bkdupchrs.L > 0 THEN % this should be changed to
        actually do a backspace word instead of backspacing all
        the way forward to the tabstop %                        09825
           BEGIN                                                09826
           astrng.L _ astrng.L - bkdupchrs.L;                   09827
```

```
      IF nlmode=fulldisp THEN                                    09828
         FOR i_1 UP UNTIL > bkdupchrs.L DO <litbc>(SP,SP);
                                                                 09829
      FOR i_1 UP UNTIL > bkdupchrs.L DO                          09830
         BEGIN                                                   09831
         *astrng* _ *astrng*, SP;                                09832
         IF nlmode=fulldisp THEN <litchr>(SP) ELSE
         !pbout(SP);                                             09833
         END;                                                    09834
      bkdupchrs.L _ 0;                                           09835
      REPEAT LOOP;                                               09836
      END;                                                       09837
   rtjtbd _ FALSE;                                               09838
   IF nlmode # fulldisp THEN echon();                            09839
   END;                                                          09840
% litstore _ <chbptr>(astrng.L) + &astrng; % %unnecessary?%
                                                                 09841
IF NOT display THEN                                              09842
   bccari _ IF <formln>(&astrng) AND NOT clpsw THEN TRUE
   ELSE FALSE;                                                   09843
% back over character echoed by tenex in teletype mode %
                                                                 09844
   IF bccari THEN <sptbbc>(137B, &astrng);                       09845
CCPOS SE(*astrng*);                                              09846
nextchar _ READC;                                                09847
CASE char _ nextchar OF                                          09848
   = ENDCHR: NULL;                                               09849
   = NP:                                                         09850
      BEGIN %space past non letters/digits%                      09851
      BUMP DOWN astrng.L;                                        09852
      IF display THEN                                            09853
         BEGIN                                                   09854
         IF NOT clpsw THEN litbc(char, nextchar _ READC);
                                                                 09855
         REPEAT CASE;                                            09856
         END;                                                    09857
      IF bccari THEN <sptbbc>(char, &astrng);                    09858
      nextchar _ READC;                                          09859
      REPEAT CASE;                                               09860
      END;                                                       09861
   ENDCASE;                                                      09862
CASE char _ nextchar OF                                          09863
   = ENDCHR: NULL;                                               09864
   = NLD:                                                        09865
      IF char = PT THEN                                          09866
         BEGIN %space past non letters/digits%                   09867
         BUMP DOWN astrng.L;                                     09868
         IF display THEN                                         09869
            BEGIN                                                09870
            IF NOT clpsw THEN litbc(char, nextchar _ READC);
                                                                 09871
            REPEAT CASE;                                         09872
            END;                                                 09873
         IF bccari THEN !pbout(8);                               09874
         nextchar _ READC;                                       09875
         REPEAT CASE;                                            09876
```

```
            END;                                                    09877
         ENDCASE;                                                   09878
      CASE char _ nextchar OF                                       09879
         = LD:                                                      09880
            BEGIN %space past letters/digits%                       09881
            BUMP DOWN astrng.L;                                     09882
            IF display AND NOT clpsw THEN litbc(char, ENDCHR )
                                                                    09883
            ELSE IF bccari THEN !pbout(8);                          09884
            nextchar _ READC;                                       09885
            REPEAT CASE;                                            09886
            END;                                                    09887
         ENDCASE;                                                   09888
      litstore _ chbptr(astrng.L) + &astrng;  %for fast character
      appends%                                                      09889
      REPEAT LOOP;                                                  09890
   (userbreak):  %user provided break character%                    09891
      IF NOT display THEN                                           09892
         BEGIN                                                      09893
         IF clpsw THEN echon();                                     09894
         IF rtjtbd THEN                                             09895
            BEGIN                                                   09896
            echon();                                                09897
            typeas($bkdupchrs);                                     09898
            END;                                                    09899
         IF slink THEN                                              09900
            BEGIN                                                   09901
            trnsloc"<^F>] _ svcfo;                                  09902
            trnsloc"<ESC>] _ sveso;                                 09903
            END;                                                    09904
         END;                                                       09905
      RETURN(2, char);                                              09906
   (rdlcacdot):  %command accept or center dot%                     09907
      IF NOT display THEN                                           09908
         BEGIN                                                      09909
         IF clpsw THEN echon();                                     09910
         IF rtjtbd THEN                                             09911
            BEGIN                                                   09912
            echon();                                                09913
            typeas($bkdupchrs);                                     09914
            END;                                                    09915
         IF slink THEN                                              09916
            BEGIN                                                   09917
            trnsloc"<^F>] _ svcfo;                                  09918
            trnsloc"<ESC>] _ sveso;                                 09919
            END;                                                    09920
         END;                                                       09921
      RETURN(1, char);                                              09922
   (rdlbs):      %backspace statement%                              09923
      *astrng* _ NULL;                                              09924
      litstore _ chbmty + &astrng;  %for fast character appends%
                                                                    09925
      IF display THEN                                               09926
         setlit()                                                   09927
      ELSE                                                          09928
         BEGIN                                                      09929
```

```
            !pbout(EOL);                                              09930
            !pbout(EOL);                                              09931
            END;                                                      09932
         REPEAT LOOP;                                                 09933
      (rdlretype):  %retype literal%                                  09934
         IF NOT clpsw THEN                                            09935
            IF display THEN                                           09936
               BEGIN                                                  09937
               rstlit();                                              09938
               aplit(&astrng);                                        09939
               END                                                    09940
            ELSE                                                      09941
               BEGIN                                                  09942
               !pbout(EOL);                                           09943
               !pbout(EOL);                                           09944
               typeas(&astrng);                                       09945
               END;                                                   09946
         REPEAT LOOP;                                                 09947
      (rdlle):      %literal escape%                                  09948
         CASE (char _ rawchr()) OF                                    09949
            = '<^F>, = '<ESC>:                                        09950
               IF slink THEN                                          09951
                  BEGIN                                               09952
                  GOTO [breaktbl[char]];                              09953
                  END                                                 09954
               ELSE REPEAT CASE( ENDCHR );                            09955
            ENDCASE                                                  010368
               IF NOT clpsw AND char NOT IN[ SP, 'z] THEN            010369
                  IF display THEN GOTO rdlspclchar                   010370
                  ELSE                                               010371
                     BEGIN                                           010372
                     &spclcharstr _ npstrad(char);                   010373
                     typeas(&spclcharstr);                           010374
                     GOTO rdladdchar;                                010375
                     END                                             010376
               ELSE GOTO [breaktbl[char]];                           010635
      (rdlspftb):   %space for tab, set flag if right justified%
                                                                      09965
         IF rtjtab THEN                                               09966
            BEGIN                                                     09967
            IF NOT display THEN                                       09968
               BEGIN % type out TNLS justified column%                09969
               IF rtjtbd THEN typeas($bkdupchrs);                     09970
               IF NOT isechon() THEN echo($" ");                      09971
               END;                                                   09972
            rtjtbd _ TRUE;                                            09973
            bkdupchrs.L _ 0;                                          09974
            echoff();                                                 09975
            END;                                                      09976
         % compute proper number of spaces and send them to terminal
         as if the user had typed them %                              09977
            IF nlmode = fulldisp THEN  %compute new
            litcolumn[litline]%                                       09978
               BEGIN                                                  09979
               spacecol _ litcolumn[litline] :=                       09980
                  MIN(fndtab(&tda, litcolumn[litline]), litrmarg);
```

```
                                                                    09981
            col _ MAX(litcolumn[litline] - spacecol, 0) + 1;        09982
            %  the following line is redundant with a mysterious
            (and inaccurate on the first tab) force that doesn't
            work with typeahead !                                   09983
            FOR i _ 1 UP UNTIL > col DO litchr(SP);                 09984
            %                                                       09985
            END                                                     09986
         ELSE                                                       09987
            BEGIN %compute column from last CR %                    09988
            col _ 0;                                                09989
            CCPOS SE(*astrng*);                                     09990
            LOOP                                                    09991
               CASE READC OF                                        09992
                  = CR,=EOL,  %compute new tab stop from this
                  column. %                                         09993
                  = ENDCHR: % this is the first line so lets
                  pretend it starts in column one.  (CR is not
                  automatically emitted after prompt for literal)
                  %                                                 09994
                     EXIT LOOP;                                     09995
                  ENDCASE                                           09996
                     BEGIN                                          09997
                     BUMP col;                                      09998
                     IF col > colmax % this should be changed to
                     the maximum line length given the level of
                     the statement. % THEN % the user is not
                     following instructions, we do the best we can
                     but dont guarantee anything %                   09999
                        BEGIN                                      010000
                        FIND SE(*astrng*) ^tp2 [CR/ENDCHR] ^tp1;
                                                                   010001
                        col _ (tp2[1] - tp1[1]) MOD (colmax-3);
                                                                   010002
                        EXIT LOOP;                                 010003
                        END;                                       010004
                     END;                                          010005
            col _ (<fndtab>(&tda,col) - col);  % determine number
            of spaces to next tab stop %                           010006
            END;                                                   010007
         FOR i _ 1 UP UNTIL >= col DO                              010008
            BEGIN                                                  010009
            ^litstore _ SP;                                        010010
            BUMP astrng.L;                                         010011
            END;                                                   010012
      IF rtjtbd THEN  % compute the number of spaces we can right
      justify %                                                    010013
         BEGIN                                                     010014
         FIND SE(*astrng*) ^tp1 $SP ^tp2;                          010015
         col _ tp1[1] - tp2[1];                                    010016
         END;                                                      010017
      REPEAT LOOP;                                                 010018
   (rdlesc): NULL;        % altmode - handle like control-F %
                                                                   010019
   (rdlctf):      % control-F %                                    010020
      remchar _ IF char = '<^F> THEN TRUE ELSE FALSE;              010021
```

```
*filstr* _ NULL;                                                    010022
FOR ind _ 1 UP UNTIL > filstr.M  DO                                 010023
   *filstr* _ *filstr*, 0;                                          010024
FIND SE(*astrng*) ^tp2                                              010025
   [ ', > CH $(SP/TAB) / ENDCHR > $(SP/TAB) ] ^tp1;                 010026
*filstr* _ tp1 tp2, '<ESC>;                                         010027
*dirstr* _ NULL;                                                    010028
IF FIND tp1 < $(SP/TAB) ', $(SP/TAB) ^dp2 (-ENDCHR AND -',)
                                                                    010029
   [ ', > CH $(SP/TAB) / ENDCHR > $(SP/TAB) ] ^dp1                  010030
      THEN *dirstr* _ + dp1 dp2, 0;                                 010031
(rdlcf1):                                                           010032
IF cdlnk AND (jfn _                                                 010033
   lgetjfn( IF dirstr.L THEN $dirstr ELSE 0,                        010034
      $filstr, $extstr, gtjidl+ (IF nwlnk THEN gtjoof ELSE
      gtjoif), $lit) ) THEN                                         010035
         BEGIN                                                      010036
         jfntostr( jfn, $dirstr, 001110B6+1);                       010037
         IF NOT SKIP !rljfn( jfn ) THEN NULL;                       010038
         *filstr*[filstr.L] _ 0;  BUMP DOWN filstr.L;               010039
         nmfnd _ exfnd _ FALSE;                                     010040
         FOR ind _ 1 UP UNTIL > filstr.L DO                         010041
            CASE *filstr*[ind] OF                                   010042
               = '.: nmfnd _ TRUE;                                  010043
               = ';: exfnd _ TRUE;                                  010044
               ENDCASE;                                             010045
         IF NOT nmfnd THEN                                          010046
            FOR ind UP UNTIL > dirstr.L DO                          010047
               IF (char _ *dirstr*[ind]) # '. THEN                  010048
                  BEGIN                                             010049
                  IF NOT rdlutl( &astrng, $char, display)
                                                                    010050
                     THEN GOTO rdlcacdot;                           010051
                  END                                               010052
               ELSE                                                 010053
                  BEGIN                                             010054
                  BUMP ind;                                         010055
                  IF NOT rdlutl( &astrng, $char, display)
                                                                    010056
                     THEN GOTO rdlcacdot;                           010057
                  IF remchar THEN REPEAT LOOP 2 ELSE EXIT
                  LOOP;                                             010058
                  END;                                              010059
         IF NOT exfnd THEN                                          010060
            FOR ind UP UNTIL > dirstr.L DO                          010061
               IF (char _ *dirstr*[ind]) # '; THEN                  010062
                  BEGIN                                             010063
                  IF NOT rdlutl( &astrng, $char, display)
                                                                    010064
                     THEN GOTO rdlcacdot;                           010065
                  END                                               010066
               ELSE                                                 010067
                  BEGIN                                             010068
                  BUMP ind;                                         010069
                  IF NOT rdlutl( &astrng, $char, display)
                                                                    010070
```

```
                        THEN GOTO rdlcacdot;                    010071
                     IF remchar THEN REPEAT LOOP 2 ELSE EXIT
                     LOOP;                                      010072
                     END;                                       010073
               FOR ind UP UNTIL > dirstr.M DO                   010074
                  IF (char _ *dirstr*[ind]) # ENDCHR THEN       010075
                     BEGIN                                      010076
                     IF NOT rdlutl( &astrng, $char, display)
                                                                010077
                        THEN GOTO rdlcacdot;                    010078
                     END                                        010079
                  ELSE                                          010080
                     BEGIN                                      010081
                     char _ ',;                                 010082
                     IF NOT rdlutl( &astrng, $char, display)
                                                                010083
                        THEN GOTO rdlcacdot;                    010084
                     EXIT LOOP;                                 010085
                     END;                                       010086
               END                                              010087
            ELSE                                                010088
               BEGIN                                            010089
               % try the alternate directory and alternate extensions%
                                                                010090
               %the following code tries each alternate directory in the
               string altdir, and tries each extension in the string
               altext for each directory. All of this madness is only
               done if the flag altdfl is true. This flag along with the
               appropiate strings are set up in by parse functions
               called in the Load Programs command.%            010091
               IF altdfl  AND FIND altd1 > [',] ^altd2 THEN     010092
                  BEGIN                                         010093
                  *dirstr* _  + altd1 altd2 ;                   010094
                  *dirstr*[dirstr.L] _ 0;                       010095
                  BUMP DOWN dirstr.L;                           010096
                  IF FIND alte1 > [',] ^alte2  THEN             010097
                     BEGIN                                      010098
                     *extstr* _ + alte1 alte2;                  010099
                     *extstr*[extstr.L] _ 0;                    010100
                     BUMP DOWN extstr.L;                        010101
                     alte1[1] _ alte2[1];                       010102
                     GOTO rdlcf1;                               010103
                     END                                        010104
                  ELSE                                          010105
                     BEGIN                                      010106
                     altd1[1] _ altd2[1] ;                      010107
                     FIND SF(*altext*) ^alte1;                  010108
                     GOTO rdlcf1;                               010109
                     END;                                       010110
                  END;                                          010111
               % if failed due to bad directory, try sans directory %
                                                                010112
                  IF cdlnk AND (r1 = $gjfx17) AND (dirstr.L := 0) THEN
                                                                010113
                     GOTO rdlcf1;                               010114
               % construct string for STDIR %                   010115
```

```
            *filstr*[filstr.L] _ 0;                                  010116
            filstr.L _ (ind _ filstr.L) - 1;                         010117
         % contruct byte pointer to directory name for STDIR %
                                                                     010118
            bytptr _ chbmty + $filstr;                               010119
         % get directory number given directory name %               010120
            IF NOT stdircall($filstr, TRUE) THEN                     010939
               BEGIN                                                 010940
               IF NOT clpsw AND NOT display THEN                     010126
                  BEGIN                                              010127
                  &spclcharstr _ npstrad(char);                      010128
                  typeas(&spclcharstr);                              010129
                  END;                                               010130
               GOTO rdladdchar;                                      010131
               END;                                                  010941
         % directory recognized ok %                                 010132
            filstr.L _ filstr.M;                                     010133
            FOR ind UP UNTIL > filstr.M DO                           010134
               IF (char _ *filstr*[ind]) THEN                        010135
                  BEGIN                                              010136
                  IF NOT rdlutl( &astrng, $char, display)            010137
                     THEN GOTO rdlcacdot;                            010138
                  END                                                010139
               ELSE                                                  010140
                  IF cdlnk THEN                                      010141
                     BEGIN                                           010142
                     char _ ",;                                      010143
                     IF NOT rdlutl( &astrng, $char, display)
                                                                     010144
                        THEN GOTO rdlcacdot;                         010145
                     EXIT LOOP;                                      010146
                     END                                             010147
                  ELSE EXIT LOOP;                                    010148
         END;                                                        010149
      REPEAT LOOP;                                                   010150
   END;                                                              010151
END.                                                                 010152
                                                                     010153
(rdlutl)     %utility for rdlit%                                      05748
   PROCEDURE (astrng, char, display);                                05749
   LOCAL STRING send[20];                                            05768
   REF astrng, char;                                                 05750
   IF astrng.L >= astrng.M THEN                                      05751
      BEGIN                                                          05752
      dismes(2, $"literal too long -- CDOT simulated");              05753
      curchr _ char _ C.;                                            05754
      RETURN( FALSE );                                               05755
      END;                                                           05756
   ^litstore _ char;                                                 05757
   BUMP astrng.L;                                                    05758
   IF NOT clpsw AND display THEN litchr(char) ELSE typech(char);
                                                                     05759
   RETURN( TRUE );                                                   05760
   END.
                                                                     05761
(formin) PROC (astrng) ;  %  accepts the address of a string. if
```

```
there is a carriage return in the last line, returns TRUE.
Otherwise, false %                                                  010154
   REF astrng;                                                      010155
   LOCAL chr, i;                                                    010156
   LOCAL TEXT POINTER tp1, tp2;                                     010157
   IF NOT spftab THEN RETURN(FALSE);                                010158
   FIND SE(*astrng*) ^tp1 [EOL/CR/LF/ENDCHR] ^tp2;                  010159
   IF tp1[1] - tp2[1] < colmax THEN RETURN(TRUE);                   010160
   RETURN(FALSE);                                                   010161
   END.
                                                                    010162
(sptbbc) PROC (char, astrng) ; % backspace TNLS carriage in highly
formatted text, 8 = Control-h:  backspaces carriage, i contains
length for astrng %                                                 010163
   LOCAL i, spclcharstr;                                            010164
   LOCAL TEXT POINTER tp1, tp2;                                     010165
   LOCAL STRING temp[2000];                                         010166
   REF spclcharstr, astrng;                                         010167
   CASE char OF                                                     010168
      IN [SP, 'z]: %normal character%                               010169
         BEGIN                                                      010170
         !pbout(8);                                                 010171
         END;                                                       010172
      = CR, = EOL, = LF: % find the preceeding line and re-type it %
                                                                    010173
         BEGIN                                                      010174
         FIND SE(*astrng*) ^tp2 < [EOL/CR/LF/ENDCHR] ^tp1;          010175
         *temp* _ EOL,EOL,tp1 tp2;                                  010176
         typeas($temp);                                             010177
         %re-position current character pointer destroyed by find %
                                                                    010178
            CCPOS SE(*astrng*);                                     010179
         END;                                                       010180
      ENDCASE %Special non-printing character%                      010181
         BEGIN                                                      010182
         &spclcharstr _ npstrad(char);                              010183
         FOR i _ 1 UP UNTIL > spclcharstr.L DO !pbout(8);           010184
         END;                                                       010185
   RETURN;                                                          010186
   END.
                                                                    010187
(sprjtb) PROC (char,astrng,bkdupchrs,col,rtjtbd) ; % entering text
after tab in right justification tab mode %                         010188
   LOCAL chr, termode, i;                                           010189
   REF bkdupchrs,astrng,rtjtbd;                                     010190
   termode _ FALSE;                                                 010339
   CCPOS SF(*rjtchr*);                                              010330
   UNTIL (chr _ READC) = ENDCHR DO                                  010331
      BEGIN                                                         010332
      IF chr = char THEN                                            010334
         BEGIN                                                      010335
         termode _ TRUE;                                            010336
         EXIT LOOP;                                                 010337
         END;                                                       010338
      END;                                                          010333
   IF termode OR (char = CR OR char = EOL OR char = LF) OR (
```

```
   bkdupchrs.L NOT< col ) THEN  % termination of right justification
   mode %                                                          010191
      BEGIN                                                        010192
      rtjtbd _ FALSE;                                              010193
      IF nlmode # fulldisp THEN echon();                           010194
      ^litstore _ char;                                            010195
      BUMP astrng.L;                                               010196
      *bkdupchrs* _ *bkdupchrs*, char;                             010197
      IF NOT clpsw THEN                                            010198
         IF nlmode # fulldisp THEN typeas(&bkdupchrs) %echo
         characters%                                               010199
         ELSE litchr(char); %echo char in literal display area%
                                                                   010200
      bkdupchrs.L_0;                                               010201
      RETURN;                                                      010202
      END;                                                         010203
    % we are in right justification mode, back up one space for TNLS,
   in DNLS, backup previous characters and enter character.  First we
   check to see if we are one space from the end of the column. %
                                                                   010204
      IF bkdupchrs.L NOT< col-2 THEN !pbout(7); % ring the bell %
                                                                   010205
      *bkdupchrs* _ *bkdupchrs*, char;                             010206
      astrng.L _ astrng.L - bkdupchrs.L;                           010207
      *astrng* _ *astrng*, *bkdupchrs*;                            010208
      IF NOT clpsw THEN                                            010209
         IF nlmode=fulldisp THEN                                   010210
            BEGIN                                                  010211
            FOR i _ 1 UP UNTIL > bkdupchrs.L DO <litbc>(char,
            ENDCHR);                                               010212
            CCPOS SF(*bkdupchrs*);                                 010213
            UNTIL (char _ READC) = ENDCHR DO <litchr>(char);       010214
            END                                                    010215
         ELSE !pbout(8);  % Control-h backspaces carriage %        010216
   RETURN;                                                         010217
   END.
                                                                   010218
%DNLS literal collection support -- put here for eficiency reasons
These routines assume literal state machinery is set up.%          03394
   (litpbout) PROCEDURE (char); %append char to literal display area%
                                                                   08259
      %-------------------%                                        08260
      LOCAL STRING send[10];                                       08261
      CASE nldevice OF                                             08262
         = devlproc:                                               08263
            CASE char OF                                           08264
               = TAB:                                              08265
                  BEGIN                                            08266
                  IF tracking THEN                                 08267
                     position(litda.daleft + litcolumn[litline]-1,
                     litda.datop + litline);                       08268
                  !bout(dspjfn, lptab);                            08269
                  IF ldspjfn THEN %linked to another devlproc%     08270
                     !bout(ldspjfn, lptab);                        08271
                  END;                                             08272
               = EOL:                                              08273
```

```
                BEGIN                                                   08274
                position(litda.daleft, litda.datop+litline+1);
                                                                        08275
                END;                                                    08276
             = CR, =LF: position(litda.daleft+litcolumn[litline]-1,
             litda.datop+litline);                                     010630
             = BC:                                                      08277
                BEGIN                                                   08278
                IF litcolumn[litline] <= 1 THEN %must back up to
                previous line and delete last char%                     08279
                   position(litda.daleft + (IF litline >= 1 THEN
                   litcolumn[litline-1] ELSE 0), litda.datop +
                   MAX(0, litline-1))                                   08280
                 ELSE IF tracking THEN                                  08281
                   position(litda.daleft + litcolumn[litline]-1,
                   litda.datop + litline);                              08282
                !bout(dspjfn, BC);                                      08283
                !bout(dspjfn, SP);                                      08284
                !bout(dspjfn, BC);                                      08285
                IF ldspjfn THEN %linked to another devlproc%            08286
                   BEGIN                                                08287
                   !bout(ldspjfn, BC);                                  08288
                   !bout(ldspjfn, SP);                                  08289
                   !bout(ldspjfn, BC);                                  08290
                   END;                                                 08291
                END;                                                    08292
             ENDCASE                                                    08293
                BEGIN                                                   08294
                IF tracking THEN                                        08295
                   position(litda.daleft + litcolumn[litline]-1,
                   litda.datop + litline);                              08296
                !bout(dspjfn, char);                                    08297
                IF ldspjfn THEN %linked to another devlproc%            08298
                   !bout(ldspjfn, char);                                08299
                END;                                                    08300
          = imlac0, = imlac1:                                           08301
             BEGIN                                                      08302
             *send* _ begmsg, 4+remfudge, apsda, litdahandle.daidr1 +
             remfudge, litdahandle.daidr2 + remfudge;                   08303
             IF char = EOL                                              08304
                THEN BEGIN                                              08305
                   *send* _ *send*, CR, LF;                             08306
                   *send*[2] _ 5+remfudge;                              08307
                   END                                                  08308
                ELSE *send* _ *send*, char;                             08309
             !sout(dspjfn, chbmty + $send, -send.L);                    08310
             IF linkcnsl THEN                                           08311
                !sout(ldspjfn, chbmty + $send, -send.L);                08312
             END;                                                       08313
          ENDCASE                                                       08314
             err ($"No Tasker");                                        08315
       RETURN;                                                          08316
       END.
                                                                        08317
(litchr) PROCEDURE (char); %append char to literal display area%
                                                                        03082
```

```
%-------------------%                                                  03083
LOCAL                                                                  03209
   spclcharstr, %address of special character string%                  03398
      %For example, <^F>, <CA>, etc.)%                                 03401
   i;           %temp: for loop control%                               03399
REF spclcharstr;                                                       03210
CASE char OF                                                           03085
   IN (SP, "z]:  %normal alpha-numeric or punctuation
   character%                                                          03174
      litpbout(char);                                                  03197
   = SP: %keep track of spaces for litline breaks%                     03108
      BEGIN                                                            03176
      spacecol _ litcolumn[litline];                                   03109
      litpbout(char);                                                  03198
      END;                                                             03177
   = EOL: %break the litline for displays%                             03111
      BEGIN                                                            03184
      spacecol _ litcolumn[litline];                                   03377
      linebreak();                                                     03378
      RETURN;                                                          04545
      END;                                                             03183
   = TAB:  %compute new litcolumn[litline] for displays%               03086
      BEGIN                                                            03182
      litpbout(TAB);                                                   03185
      spacecol _ litcolumn[litline] :=                                 03089
         MIN(fndtab(&tda, litcolumn[litline])-1, litrmarg);
                                                                       03090
      ^tabstore _ i _ MAX(litcolumn[litline] - spacecol, 0);
                                                                       03091
      FOR i DOWN UNTIL <= 0 DO litpbout(3);  %non-printing fill
      character%                                                       03092
      END;                                                             03181
   = LF: %break the litline for displays%                              03095
      BEGIN                                                            03180
      spacecol _ litcolumn[litline];                                   03098
      litline _ MIN(litline+1, 34);                                    03099
      litsup(littop + (litline+1)*litvinc);                            03100
      litcolumn[litline] _ spacecol;                                   03101
      litpbout(LF);                                                    03200
      END;                                                             03178
   = CR: %break the litline for displays%                              03104
      BEGIN                                                            03179
      litcolumn[litline] _ spacecol _ 1;                               03106
      litpbout(CR);                                                    03105
      RETURN;                                                          04546
      END;                                                             03107
   ENDCASE  %Special non-printing character%                           02891
      BEGIN                                                            03385
      %get string which represents special char%                       02892
         &spclcharstr _ npstrad(char);                                 02893
      spacecol _ litcolumn[litline];                                   03208
      IF litcolumn[litline] + spclcharstr.L > litrmarg THEN
                                                                       02894
         linebreak();                                                  02896
      IF spclcharstr.L THEN                                            04261
         BEGIN                                                         04263
```

```
            CCPOS SF(*spclcharstr*);                                    03186
            CASE char _ READC OF                                        03187
               = ENDCHR: BUMP DOWN litcolumn[litline];                  03188
               ENDCASE                                                  03189
                  BEGIN                                                 03193
                  litpbout(char);                                       03192
                  BUMP litcolumn[litline];                              04260
                  REPEAT CASE;                                          03190
                  END;                                                  03191
            END;                                                        04262
         END;                                                           02902
   IF (litcolumn[litline] _ litcolumn[litline] + 1) > litrmarg
   THEN %must break litline%                                            03150
      BEGIN                                                             03205
      BUMP DOWN litcolumn[litline];                                     03153
      linebreak();                                                      03203
      END;                                                              03206
   RETURN;                                                              03172
   END.
                                                                        03173
(litbc) PROCEDURE (char, prevchar);     %backspace character in
literal area%                                                          010220
   LOCAL i, spclcharstr;                                               010221
   REF spclcharstr;                                                    010222
   litpbout(BC);                                                       010223
   CASE char OF                                                        010224
      IN [SP,"z]:  %normal character%                                  010225
         IF (litcolumn[litline] _ litcolumn[litline] -1) < 1 THEN
                                                                       010226
            %go to previous litline%                                   010227
            IF litline = 0 THEN litcolumn[litline] _ 1                 010228
            ELSE                                                       010229
               BEGIN                                                   010230
               litline _ litline-1;                                    010231
               spacecol _ 1;                                           010232
               CASE prevchar OF                                        010233
                  = EOL, = LF, = CR: NULL;                             010234
                  = ENDCHR: NULL;                                      010235
                  ENDCASE                                              010236
                     BEGIN                                             010237
                     litpbout(BC); litpbout(prevchar);                 010238
                        %necessary because of the way sequential
                        display areas treats EOL, LF, and CR%
                                                                       010239
                     END;                                              010240
               END;                                                    010241
         = TAB:                                                        010242
            BEGIN                                                      010243
            i _ .tabstore;                                             010244
            FOR i DOWN UNTIL <= 0 DO litpbout(BC);                     010245
            i _ .tabstore + 1;                                         010246
            %back up tabstore%                                         010247
               r1 _ tabstore;                                          010248
               r2 _ tabstore-1;                                        010249
               UNTIL r2 = r1 DO                                        010250
                  BEGIN                                                010251
```

```
            r3 _ r2;                                                    010252
            !IBP r2;                                                    010253
            END;                                                        010254
         tabstore _ r3;                                                 010255
      IF (litcolumn[litline] _ litcolumn[litline] - i) < 1 THEN
                                                                        010256
         %go to previous litline%                                       010257
         IF litline = 0 THEN litcolumn[litline] _ 1                     010258
         ELSE                                                           010259
            BEGIN                                                       010260
            litline _ litline-1;                                        010261
            spacecol _ 1;                                               010262
            CASE prevchar OF                                            010263
               = EOL, = LF, = CR: NULL;                                 010264
               = ENDCHR: NULL;                                          010265
               ENDCASE                                                  010266
                  BEGIN                                                 010267
                  litpbout(BC); litpbout(prevchar);                     010268
                     %necessary because of the way sequential
                     display areas treats EOL, LF, and CR%
                                                                        010269
                  END;                                                  010270
            END;                                                        010271
      END;                                                              010272
   = EOL, = LF:                                                         010273
      BEGIN                                                             010274
      IF litline = 0 THEN litcolumn[litline] _ 1                        010275
      ELSE                                                              010276
         BEGIN                                                          010277
         litline _ litline-1;                                           010278
         spacecol _ 1;                                                  010279
         % commented out 11-DEC-76 by KJM                              010632
         CASE prevchar OF                                               010280
            = EOL, = LF, = CR: NULL;                                    010281
            = ENDCHR: NULL;                                             010282
            ENDCASE                                                     010283
               BEGIN                                                    010284
               litpbout(prevchar);                                      010285
                  %%necessary because of the way sequential
                  display areas treats EOL, LF, and CR%% 010286
               END;                                                     010287
         %                                                              010631
         END;                                                           010288
      END;                                                              010289
   = CR:  %does this code make sense?  KJM%                             010290
      BEGIN                                                             010291
      CASE prevchar OF                                                  010292
         = EOL, = LF, = CR: NULL;                                       010293
         = ENDCHR: NULL;                                                010294
         ENDCASE                                                        010295
            BEGIN                                                       010296
            litpbout(BC); litpbout(prevchar);                           010297
               %necessary because of the way sequential display
               areas treat CR%                                          010298
            END;                                                        010299
      END;                                                              010300
```

```
          ENDCASE  %non-printing, special character%                    010301
             BEGIN                                                      010302
             &spclcharstr _ npstrad(char);                              010303
                %get address of string which represents this
                character%                                              010304
             FOR i _ spclcharstr.L-1 DOWN UNTIL <=0 DO                  010305
                litpbout(BC);                                           010306
             IF (litcolumn[litline] _ litcolumn[litline] -
             spclcharstr.L) < 1 THEN                                    010307
                %go to previous litline%                                010308
                IF litline = 0 THEN litcolumn[litline] _ 1              010309
                ELSE                                                    010310
                   BEGIN                                                010311
                   litline _ litline-1;                                 010312
                   spacecol _ 1;                                        010313
                   CASE prevchar OF                                     010314
                      = EOL, = LF, = CR: NULL;                          010315
                      = ENDCHR: NULL;                                   010316
                      ENDCASE                                           010317
                         BEGIN                                          010318
                         litpbout(BC); litpbout(prevchar);              010319
                            %necessary because of the way sequential
                            display areas treats EOL, LF, and CR%
                                                                        010320
                         END;                                           010321
                   END;                                                 010322
             END;                                                       010323
          CASE prevchar OF                                              010324
             =NP, =ENDCHR: spacecol _ litcolumn[litline] - 1;           010325
             ENDCASE IF spacecol >= litcolumn[litline] THEN spacecol _ 1;
                                                                        010326
          RETURN;                                                       010327
          END.                                                          010328
                                                                        010329
%...Typewriter NLS output routines...%                                    0394
   (todco) PROCEDURE (char);                                              0400
      %output a character to tty if feedbk > 0%                           0401
      %-----------------%                                                 0402
      IF feedbk > 0 THEN typech(char);                                    0403
      RETURN;                                                             0404
      END.
                                                                          0405
   (typectl) %type (translated) character on tty - prints special chars%
                                                                         04769
      PROCEDURE(char);                                                   04786
      LOCAL tchar, spclcharstr;                                          04770
      REF spclcharstr;                                                   04771
      %-----------------%                                                04772
      tchar _ trnslo[char];                                              04773
      CASE tchar OF                                                      04774
         = EOL,                                                          09661
         = nullch,                                                       04787
         = 'V-100B : NULL;                                               04788
         = TAB,                                                          04789
         IN [SP, 'z]: !pbout(tchar);                                     04778
         ENDCASE                                                         04779
```

```
        BEGIN                                                    04780
        &spclcharstr _ npstrad(tchar);                           04781
        typeas(&spclcharstr);                                    04782
        END;                                                     04783
   RETURN                                                        04784
   END.
                                                                 04785
(echo) %echo a string%                                            0452
   %This procedure is used for command echoing. It is the same as
   TYPEAS, except that it observes the limit set in feebk. It expects
   to be passed the address of the string to be echoed.  It types
   characters from the string onto the tty until either          0453
      (a) the entire string has been typed out, or               0454
      (b) the number of characters typed is equal to the value   0455
      in feedbk.%                                                0456
   %-----------------%                                           0457
   PROCEDURE (string);                                           0458
   LOCAL                                                         0459
      count,  %count for string length%                          0460
      bytptr; %byte pointer for the current character%           0461
   REF string;                                                   0462
   IF string.L <= feedbk THEN typeas(&string)                    0463
   ELSE                                                          0464
      BEGIN                                                      0465
      bytptr _ chbptr(count _ empty) + &string;                  0466
      UNTIL (count _ count + 1) > feedbk DO                      0467
         BEGIN                                                   0468
         r1 _ ^bytptr;                                           0469
         !JSYS pbout;                                            0470
         END;                                                    0471
      END;                                                       0472
   RETURN;                                                       0473
   END.
                                                                 0474
(prompt) PROC(astr);                                             0475
   %Prompt user by typing astr%                                  0476
   typeas(astr);                                                 0477
   RETURN END.
                                                                 0478
(typebuff)PROC;                                                  0712
   %Type out the contents of the look ahead input buffer%        0713
   LOCAL i;                                                      0714
   IF feedbk > 0 THEN                                           03455
      BEGIN                                                     03456
      i _ buffs;                                                 0715
      UNTIL i = buffn DO                                         0716
         BEGIN                                                   0717
            typectl(buff[i]); % type out next char %             0718
            IF (i _ i + 1) > buffsz % bump buffer index %        0719
               THEN i _ 0;                                       0720
         END;                                                    0721
      END;                                                      03457
   RETURN;                                                       0722
   END.                                                          0723
%...Sending a message to another terminal....%                   04790
(xsndtmsg) PROC (messtring, userst);                             05973
```

```
    % given a message and user directory name, xsndmsg finds all the
    places that user is currently logged in and attached and sends the
    message. The job MUST BE enabled for this to work.  Otherwise you
    get an illegal instruction trap.  It returns TRUE/FALSE.%
                                                                    05974
    LOCAL retflg, i, dirno;                                         05975
    LOCAL STRING outstr[100];                                       05976
    REF userst, messtring;
                                                                    05977
    % Append 0 to user string for TENEX stdir JSYS %                05978
       *outstr* _ *userst*, 0;                                      05979
    % convert user string to directory number %                     05980
       !stdir(TRUE, $outstr + chbmty);                              05981
       GOTO xsndbad;                                                05982
       GOTO xsndbad;                                                05983
       dirno _ r1.RH;                                               05984
       retflg _ FALSE;                                              05985
       IF messtring.L > 99 THEN messtring.L _ 99;                   05986
       *outstr* _ EOL, *messtring*;                                 05987
    % look for match(es) on this user - currently logged in and
    attached %                                                      05988
       FOR i _ 1 UP UNTIL > 778 DO                                  05989
          BEGIN                                                     05990
          r1.LH _ i;  % job # %                                     05991
          r1.RH _ jobtbl; %global%                                  05992
          IF NOT SKIP !getab(r1) THEN RETURN(retflg); %probably end of
          table%                                                    05993
          IF dirno = r1.RH THEN retflg _ csndtms($outstr, i);       05994
          END;                                                      05995
       RETURN(retflg);                                              05996
    (xsndbad):                                                      05997
       RETURN(FALSE);                                               05998
    RETURN END.
                                                                    05999
(csndtms) PROC (messtring, jobno); %core send tty messtring%       04812
    % Given a message and a TENEX job number, this routine sends it if
    the job is attached. The job number is assumed to be valid; it
    returns FALSE if detached, TRUE if sent ok %
                                                                    04825
    LOCAL ttyno;  %controlling teletype number%                     04813
    REF messtring;  %being sent out%
                                                                    04814
    r1.LH _ jobno;                                                  04815
    r1.RH _ ttytbl;  %global%                                       04816
    % get tty number %                                             04817
       IF NOT SKIP !getab(r1) THEN RETURN(FALSE);                   04818
       IF (ttyno _ r1) = -1 THEN RETURN(FALSE); %detached%          04819
       !sout(485 .V ttyno.LH, &messtring + chbmty, -messtring.L);
                                                                    04820
       RETURN(TRUE);                                                04821
    END.
                                                                    04822
%...Monitor echoing control...%                                     01415
   (echon) PROCEDURE;                                               0724
      %echo each character with itself; each character is a break
      character%                                                    0725
```

```
   %-----------------%                                                  0726
   IF feedbk # 0 THEN                                                  0727
      BEGIN                                                            0728
      r1 _ 18M; % controlling tty %                                    0729
      !JSYS rfmod; % read mode %                                       0730
      r2 _ r2 .A 777777001777B;                                        0731
      r2 _ r2 .V 174B3; %break on all, echo on%                        0732
         %change to not break on alpha-numeric%                       0733
      !JSYS sfmod;                                                     0734
      echofg _ TRUE;                                                   0735
      END;                                                             0736
   RETURN;                                                             0737
   END.
                                                                       0738
(echoff) PROCEDURE;                                                    0757
   %no echo for any character; break on all characters%                0758
   %-----------------%                                                 0759
   r1 _ 18M; % controlling tty %                                       0760
   !JSYS rfmod; % read mode %                                          0761
   r2 _ r2 .A 777777001777B;                                           0762
   r2 _ r2 .V 17B4; %break on all, echo off%                           0763
   !JSYS sfmod;                                                        0764
   echofg _ FALSE;                                                     0765
   RETURN END.
                                                                       0766
(isechon) PROCEDURE;                                                   0739
   %returns true if echoing on %                                       0740
   %-----------------%                                                 0741
   r1 _ 18M; % controlling tty %                                       0742
   !JSYS rfmod; % read mode %                                          0743
   RETURN(r2 .A 4B3);                                                  0744
   END.
                                                                       0745
(echolt) PROCEDURE;                                                    0767
   %echo all characters(immediate or deferred), break on all but
   alpha-numerics%                                                     0768
   %-----------------%                                                 0769
   r1 _ 18M; % controlling tty %                                       0770
   !JSYS rfmod; % read mode %                                          0771
   r2 _ r2 .A 777777001777B;                                           0772
   r2 _ r2 .V 164B3;                                                   0773
   !JSYS sfmod;                                                        0774
   echofg _ TRUE;                                                      0775
   RETURN;                                                             0776
   END.
                                                                       0777
(echoimed) PROCEDURE;                                                  0746
   %echo each character with itself; each character is a break
   character%                                                          0747
   %-----------------%                                                 0748
   r1 _ 18M; % controlling tty %                                       0749
   !JSYS rfmod; % read mode %                                          0750
   r2 _ r2 .A 777777001777B;                                           0751
   r2 _ r2 .V 172B3; %break on all, echo on%                           0752
   !JSYS sfmod;                                                        0753
   echofg _ TRUE;                                                      0754
```

```
   RETURN;                                                          0755
   END.
                                                                    0756
%...bug marks, cursor, CFL, name area, view specs, date-time...%
                                                                    0778
   %...bug marks...%                                                01416
   (markit) PROCEDURE (da, xcrd, ycrd, char);                       03674
      LOCAL bmlsrt[12], lsid, daid, x, y;                           03675
      REF da;                                                       03676
      BUMP bmcnt;                                                   03677
      CASE nldevice OF                                              03678
         = devlproc:                                                03679
            BEGIN                                                   03680
            lpmarkit(x _ xcrd+da.daleft, y _ ycrd+da.datop);        03681
            PUSH char ON bugmarks;                                  06186
            PUSH x ON bugmarks;                                     06187
            PUSH y ON bugmarks;                                     06188
            END;                                                    03682
         ENDCASE                                                    03683
            BEGIN                                                   03684
            bmlsrt.rtbps _ chbptr(empty) + $bm;                     03685
            bmlsrt.rtbpe _ chbptr(bm.L) + $bm;                      03686
            bmlsrt.rtexis _ TRUE;                                   03687
            bmlsrt.rtcsize _ da.dacsize;                            03688
            bmlsrt.rtfont _ da.dafont;                              03689
            bmlsrt.rthinc _ da.dahinc;                              03690
            bmlsrt.rtlsid _ 0;                                      03691
            bmlsrt.rtx1 _ xcrd;                                     03692
            bmlsrt.rty _ ycrd;                                      03693
            wrtstr(&da, $bmlsrt);                                   03694
            daid _ da.dahandle;                                     03696
            PUSH daid ON bugmarks;                                  03697
            daid _ da.dalhandle;                                    03698
            PUSH daid ON bugmarks;                                  03699
            lsid _ bmlsrt.rtlsid;                                   03700
            PUSH lsid ON bugmarks;                                  03701
            END;                                                    03695
      RETURN;                                                       03702
      END.
                                                                    03703
   (bmoff) PROCEDURE; %remove all bug marks%                        08318
      %This procedure removes all of the marks (currently circles)
      that provide feedback of bug selects on the screen.%          08319
      %----------------%                                           08320
      LOCAL lsid, daid, ldaid, save, bmempty, char, x, y;           08321
      LOCAL STRING send[20];                                        08322
      IF NOT bmcnt THEN RETURN;  %no bug marks on screen%           08323
      save _ bugmarks;                                              08324
      RESET bugmarks;                                               08325
      bmempty _ bugmarks := save;                                   08326
      CASE nldevice OF                                              08327
         = devlproc:                                                08328
            BEGIN                                                   08329
            UNTIL bugmarks = bmempty DO                             08330
               BEGIN                                                08331
               lppopmark();                                         08332
```

```
            POP bugmarks TO y;                                          08333
            POP bugmarks TO x;                                          08334
            POP bugmarks TO char;                                       08335
            *send* _ char;                                              08336
            lpdspstr(x, y, $send, FALSE);                               08338
            END;                                                        08339
         END;                                                           08340
      = imlac0, = imlac1:                                               08341
         BEGIN                                                          08342
         UNTIL bugmarks = bmempty DO                                    08343
            BEGIN                                                       08344
            POP bugmarks TO lsid;                                       08345
            POP bugmarks TO ldaid;                                      08346
            POP bugmarks TO daid;                                       08347
            *send* _ begmsg, 5+remfudge, remssda,
            daid.daidr1+remfudge, daid.daidr2+remfudge, lsid,
            TRUE;                                                       08348
            !sout(dspjfn, chbmty + $send, -send.L);                     08349
            IF ldaid THEN                                               08350
               !sout(ldspjfn, chbmty + $send, -send.L);                 08351
            END;                                                        08352
         END;                                                           08353
      ENDCASE                                                           08354
         err($"No Tasker");                                             08355
   bmcnt _ 0;                                                           08356
   RETURN;                                                              08357
   END.
                                                                        08358
(delbm) PROCEDURE; %turn the most recent bug mark off%                  08359
   %if the most recent bug  was via a marker, or if there are no
   bug marks on the screen, this routine just returns%                  08360
   %------------------%                                                08361
   LOCAL lsid, daid, ldaid, save, bmempty, char, x, y;                  08362
   LOCAL STRING send[20];                                               08363
   IF NOT bmcnt THEN RETURN;  %no bug marks on screen%                  08364
   save _ bugmarks;                                                     08365
   RESET bugmarks;                                                      08366
   bmempty _ bugmarks := save;                                          08367
   CASE nldevice OF                                                     08368
      = devlproc:                                                       08369
         IF bugmarks NOT= bmempty THEN                                  08370
            BEGIN                                                       08371
            lppopmark();                                                08372
            POP bugmarks TO y;                                          08373
            POP bugmarks TO x;                                          08374
            POP bugmarks TO char;                                       08375
            *send* _ char;                                              08376
            lpdspstr(x, y, $send, FALSE);                               08378
            END;                                                        08379
      = imlac0, = imlac1:                                               08380
         IF bugmarks NOT= bmempty THEN                                  08381
            BEGIN                                                       08382
            POP bugmarks TO lsid;                                       08383
            POP bugmarks TO ldaid;                                      08384
            POP bugmarks TO daid;                                       08385
            *send* _ begmsg, 5+remfudge, remssda,
```

```
            daid.daidr1+remfudge, daid.daidr2+remfudge, lsid, 1;
                                                                    08386
            !sout(dspjfn, chbmty + $send, -send.L);                 08387
            IF ldaid THEN                                           08388
               !sout(ldspjfn, chbmty + $send, -send.L);             08389
            END;                                                    08390
         ENDCASE                                                    08391
            err($"No Tasker");                                      08392
      BUMP DOWN bmcnt;                                              08393
      RETURN;                                                       08394
      END.
                                                                    08395
%...Bug Selection Routines...%
                                                                    02354
   (pbug) PROCEDURE (cordinates);  %process a bug selection%        02355
      %This routine converts a word of 18-bit universal display     02356
      cordinates into a t-pointer (stid,character count) using      02357
      information in the line segment reference table (LSRT) of     02358
      the appropriate text area diplay area.%                       02359
      %----------------%                                           02360
      LOCAL                                                         02361
         da, ycrdnt, xcrdnt, stid, chrcnt, char;                    02362
      LOCAL TEXT POINTER z1;                                        06191
      REF da;                                                       02363
      IF NOT (&da _ dsparea(findda(cordinates))) THEN err($"display
      error, pbug");                                                02364
      ycrdnt _ MAX(0, cordinates.ycord - da.datop - da.davinc/2);
                                                                    02365
      xcrdnt _ MAX(0, cordinates.xcord - da.daleft);                02366
      stid _ fndchr(&da, xcrdnt, ycrdnt : chrcnt, xcrdnt, ycrdnt);
                                                                    02367
         %find the closest selectable character, and get            02368
         t-pointer%                                                 02369
      %mark the character%                                          02370
         IF putbackchar THEN                                        06181
            BEGIN %save the character if device is a line-processor %
                                                                    06182
            z1 _ stid; z1[1] _ chrcnt;                              06165
            FIND z1;                                                06167
            char _ READC;                                           06184
            END                                                     06183
         ELSE char _ SP;                                            06189
         markit(&da, xcrdnt, ycrdnt, char);                         02371
      RETURN(stid, chrcnt);                                         02372
      END.                                                          02373
                                                                    02374
   (fndchr) PROCEDURE (da, xcrdnt, ycrdnt);  %find character
   position%                                                        02375
      %Given display area entry address and 18-bit cordinates,      02376
      this routine finds the nearest charater, returns te stid,     02377
      character count, and cordinates of that character.%           02378
      %----------------%                                           02379
      LOCAL base,  diff, chrcnt, target, min, ls, ccnt, end;        02380
      REF ls, da;                                                   02381
      %find nearest line with ycrdnt%                               02382
         &ls _ da.dalsrt;                                           02383
```

```
      end _ da.dalsz * lsrtl + da.dalsrt;                              02384
      target _ 0;                                                      02385
      min _ bmarg-tmarg;                                               02386
      DO                                                               02387
         IF ls.rtexis AND ls.rtsrce = srcstat AND (diff _
         MAX(ls.rty-ycrdnt, ycrdnt-ls.rty)) < min THEN                 02388
            BEGIN                                                      02389
            min _ diff;                                                02390
            target _ &ls;                                              02391
            END                                                        02392
      UNTIL (&ls _ &ls + lsrtl) >= end;                                02393
      IF NOT target THEN err($"pbug error, fndchr");                   02394
      &ls _ target;                                                    02395
      ycrdnt _ ls.rty;                                                 02396
   %find all line segments within da.davinc of ycrdnt%                 02397
      base _ pbugpt _ $pbugtb;                                         02398
      &ls _ da.dalsrt;                                                 02399
      ccnt _ 1;                                                        02400
      UNTIL &ls >= end OR pbugpt > $pbugte DO                          02401
         BEGIN                                                         02402
         IF ls.rtexis AND ls.rtsrce = srcstat AND                      02403
         ls.rty IN [ycrdnt-da.davinc, ycrdnt+da.davinc] THEN           02404
            [pbugpt := pbugpt+1] _ &ls;                                02405
         &ls  _ &ls + lsrtl;                                           02406
         END;                                                          02407
   %get nearest character%                                             02408
      end _ pbugpt;                                                    02409
      min _ (rmarg-lmarg)*(rmarg-lmarg);                               02410
      target  _ ccnt _ 0;                                              02411
      UNTIL base >= end DO                                             02412
         BEGIN                                                         02413
         &ls _ [base := base+1];                                       02414
         IF xcrdnt IN [ls.rtx1, ls.rtx2] THEN                          02415
            DIV (xcrdnt - ls.rtx1)/ls.rthinc, chrcnt, diff             02416
         ELSE                                                          02417
            IF (diff _ ls.rtx1 - xcrdnt) > 0 THEN                      02418
               chrcnt _ 0                                              02419
            ELSE                                                       02420
               BEGIN                                                   02421
               diff _ xcrdnt -ls.rtx2;                                 02422
               chrcnt _ (ls.rtx2 - ls.rtx1)/ls.rthinc;                 02423
               END;                                                    02424
         IF (diff _ diff*diff + (ls.rty-ycrdnt)*(ls.rty-ycrdnt)) <
         min THEN                                                      02425
            BEGIN                                                      02426
            min _ diff;                                                02427
            ccnt _ chrcnt;                                             02428
            target _ &ls;                                              02429
            END;                                                       02430
         END;                                                          02431
      IF NOT (&ls _ target) THEN err($"no hits in fndchr");            02432
   RETURN(ls.rtstid, ls.rtccnt + ccnt/ls.rtcbug,
   ls.rtx1+ccnt*ls.rthinc, ls.rty)                                     02433
   END.                                                                02434
                                                                       02435
%...cursor...%                                                         01417
```

```
   (arm) PROCEDURE;  %arm cursor%                                    01253
       RETURN;                                                       04827
      IF NOT (csrarmd := TRUE) THEN csrstr($rmdcr, tacsize, 0, 0);
                                                                     01254
       RETURN;                                                       01255
       END.
                                                                     01256
   (disarm) PROCEDURE;  %disarm cursor%                              01257
      RETURN;                                                        04826
      IF csrarmd := FALSE THEN csrstr($drmdcr, tacsize, 0, 0);       01258
      RETURN;                                                        01259
      END.
                                                                     01260
   (csrstr) PROCEDURE (string, csize, hinc, font); %set cursor
   string%                                                           08396
      %The string that tracks the mouse position is set by this
      routine %                                                      08397
      %----------------%                                            08398
      LOCAL bp;                                                      08399
      LOCAL STRING send[30];                                         08400
      REF string;                                                    08401
      CASE nldevice OF                                               08402
         = devlproc: NULL;                                           08403
         = imlac0, = imlac1:                                         08404
            BEGIN                                                    08405
            *send* _ begmsg, 5+string.L+remfudge, remscsr,
            string.L+1, csize, hinc, font, *string*;                 08406
            !sout(dspjfn, chbmty + $send, -send.L);                  08407
            IF linkcnsl THEN                                         08408
               !sout(ldspjfn, chbmty + $send, -send.L);              08409
            END;                                                     08410
         ENDCASE                                                     08411
            err($"No Tasker");                                       08412
      RETURN;                                                        08413
      END.
                                                                     08414
%...viewspec area...%                                                01418
   (dspvsp) PROCEDURE                                                03815
      (vspec1,    % 1st word of view specs %                         03816
       vspec2,    % 2nd word of view specs %                         03817
       which);    % =1 -- LT "numbers". = 2 -- view specs, =3 -- both
      %                                                              03818
      %sets up strings VSSTR1 (L and T) and VSSTR2 (hjuCP)%          03819
      %----------------%                                            03820
      LOCAL llsrt, firstd, secndd;                                   03821
      REF llsrt;                                                     03822
      %exit if request same as current viewspecs%                    03823
      IF (vspec1 = vspsav) AND (vspec2 = vspsav[1]) THEN RETURN;
                                                                     03824
      (vspsav,vspsav[1]) _ (vspec1,vspec2);                          03825
      IF which .A 1 THEN                                             03826
         BEGIN                                                       03827
         IF vspec1.vsrlev THEN                                       03828
            BEGIN                                                    03829
            *vsstr1* _ 'R;                                           03830
            IF vspec1.vslevd THEN *vsstr1* _ *vsstr1*, '-            03831
```

```
      ELSE *vsstr1* _ *vsstr1*, '+;                                    03832
      *vsstr1* _ *vsstr1*, STRING(vspec1.vslev), SP;                   03833
      END                                                              03834
   ELSE                                                                03835
      CASE vspec1.vslev OF                                             03836
         >= 63:   *vsstr1* _ "ALL ";                                   03837
         <= 9:    *vsstr1* _ "  ", vspec1.vslev + '0, SP;              03838
         ENDCASE                                                       03839
            BEGIN                                                      03840
            DIV vspec1.vslev / 10, firstd, secndd;                     03841
            *vsstr1* _ SP, firstd + '0, secndd + '0, SP;               03842
            END;                                                       03843
   CASE vspec1.vstrnc OF                                               03844
      >= 63:   *vsstr1* _ *vsstr1*, "ALL";                             03845
      <= 9:    *vsstr1* _ *vsstr1*, "  ", vspec1.vstrnc + '0;
                                                                       03846
      ENDCASE                                                          03847
         BEGIN                                                         03848
         DIV vspec1.vstrnc / 10, firstd, secndd;                       03849
         *vsstr1* _ *vsstr1*, SP, firstd + '0, secndd + '0;
                                                                       03850
         END;                                                          03851
   &llsrt _ ltvsda.dalsrt;                                             03852
   CASE nldevice OF                                                    03853
      = devlproc:                                                      03854
         BEGIN                                                         03855
         UNTIL vsstr1.L >= llsrt.rtx2 - llsrt.rtx1  DO                 04249
            *vsstr1* _ *vsstr1*, SP;                                   04250
         lpdspstr(llsrt.rtx1+ltvsda.daleft,
         llsrt.rty+ltvsda.datop, $vsstr1, NOT ltsmall);                03856
         END;                                                          03857
      ENDCASE                                                          03858
         BEGIN                                                         03859
         llsrt.rtbps _ chbptr(empty) + $vsstr1;                        03860
         llsrt.rtbpe _ chbptr(vsstr1.L) + $vsstr1;                     03861
         wrtstr (&ltvsda, &llsrt);                                     03862
         END;                                                          03863
   END;                                                                03864
IF which .A 2 THEN                                                     03865
   BEGIN                                                               03866
   *vsstr2* _                                                          03867
      IF vspec1.vsbrof THEN 'g ELSE IF vspec1.vsplxf THEN 'l
      ELSE 'h,                                                         03868
      IF vspec1.vscapf THEN 'i ELSE IF vspec1.vscakf THEN 'k
      ELSE 'j,                                                         03869
      IF vspec1.vsdaft THEN 'u ELSE 'v,                                03870
      IF vspec1.vsnamf THEN 'C ELSE 'D,                                03871
      IF vspec1.vsusqf THEN 'O ELSE 'P;                                03872
   &llsrt _ vspcda.dalsrt;                                             03873
   CASE nldevice OF                                                    03874
      = devlproc:                                                      03875
         BEGIN                                                         03876
         UNTIL vsstr2.L >= llsrt.rtx2 - llsrt.rtx1 DO                  04247
            *vsstr2* _ *vsstr2*, SP;                                   04248
         lpdspstr(llsrt.rtx1+vspcda.daleft,
         llsrt.rty+vspcda.datop, $vsstr2, NOT ltsmall);                03877
```

```
            END;                                                    03878
         ENDCASE                                                    03879
            BEGIN                                                   03880
            llsrt.rtbps _ chbptr(empty) + $vsstr2;                  03881
            llsrt.rtbpe _ chbptr(vsstr2.L) + $vsstr2;               03882
            wrtstr (&vspcda, &llsrt);                               03883
            END;                                                    03884
      END;                                                          03885
   RETURN;                                                          03886
   END.
                                                                    03887
(lti) PROCEDURE; %set LT viewspec area large%                       03888
   %Several characters in the upper left corner of the screen
   provide feedback of the current view specs.  Calling this
   routine will cause the display of these characters standout.%
                                                                    03889
   %----------------%                                              03890
   LOCAL llsrt;   REF llsrt;                                        03891
   CASE nldevice OF                                                 03892
      = devlproc:                                                   03893
         IF ltsmall := FALSE THEN                                   03894
            BEGIN                                                   03895
            lpdspstr(ltvsda.daleft, ltvsda.datop, $vsstr1, TRUE);
                                                                    03896
            lpdspstr(vspcda.daleft, vspcda.datop, $vsstr2, TRUE);
                                                                    03897
            END;                                                    03898
      ENDCASE                                                       03899
         IF ltsmall := FALSE THEN                                   03900
            BEGIN                                                   03901
            &llsrt _ ltvsda.dalsrt;                                 03902
            llsrt.rtcsize _ 2;                                      03903
            llsrt.rthinc  _ 5120;                                   03904
            llsrt.rtbps   _ -1;   %rewrite same string%             03905
            wrtstr (&ltvsda, &llsrt);                               03906
            &llsrt _ vspcda.dalsrt;                                 03907
            llsrt.rtcsize _ 2;                                      03908
            llsrt.rthinc  _ 5120;                                   03909
            llsrt.rtbps   _ -1;   %rewrite same string%             03910
            wrtstr (&vspcda, &llsrt);                               03911
            END;                                                    03912
   RETURN;                                                          03913
   END.
                                                                    03914
(lts) PROCEDURE; %set LT viewspec area small%                       03915
   %Calling this routine will cause the display of the view spec
   feedback area to not stand out.%                                 03916
   %----------------%                                              03917
   LOCAL llsrt;   REF llsrt;                                        03918
   CASE nldevie OF                                                  03919
      = devlproc:                                                   03920
         IF NOT (ltsmall := TRUE) THEN                              03921
            BEGIN                                                   03922
            position(ltvsda.daleft, ltvsda.datop);                  03923
            endstandout();                                          04206
            lpdspstr(current, current, $vsstr1, FALSE);             04205
```

```
            position(vspcda.daleft, vspcda.datop);                  03924
            endstandout();                                          04208
            lpdspstr(current, current, $vsstr2, FALSE);             04207
            END;                                                    03925
         ENDCASE                                                    03926
            IF NOT (ltsmall := TRUE) THEN                           03927
            BEGIN                                                   03928
            &llsrt _ ltvsda.dalsrt;                                 03929
            llsrt.rtcsize _ ltvsda.dacsize;                         03930
            llsrt.rthinc  _ ltvsda.dahinc;                          03931
            llsrt.rtbps   _ -1;   %rewrite same string%             03932
            wrtstr (&ltvsda, &llsrt);                               03933
            &llsrt _ vspcda.dalsrt;                                 03934
            llsrt.rtcsize _ vspcda.dacsize;                         03935
            llsrt.rthinc  _ vspcda.dahinc;                          03936
            llsrt.rtbps   _ -1;   %rewrite same string%             03937
            wrtstr (&vspcda, &llsrt);                               03938
            END;                                                    03939
      RETURN                                                        03940
      END.
                                                                    03941
%...name area...%                                                   01419
   (dn) PROCEDURE (astrng); %display string in name area%           03942
      %An A-string is displayed in the name area by calling dn with
      the address of the A-string.%                                 03943
      %IF the global NAMERESET is TRUE then there is nothing being
      displayed in the name area%                                   03944
      %----------------%                                           03945
      LOCAL llsrt, chrcnt, npstring, char, olength;                 03947
      LOCAL STRING tmpstr[100];                                     06202
      REF llsrt, astrng, npstring;                                  03948
      REF  rawchr;                                                  06194
      REF litda, msgda, tda, cflda, ltvsda, vspcda, namda, subda;
                                                                    06195
      IF namereset AND astrng.L = empty THEN RETURN;                03949
      namereset _ IF astrng.L THEN FALSE ELSE TRUE;                 03950
      IF nlmode = typewriter THEN                                   06207
         BEGIN                                                      06208
         CASE astrng.L OF                                           06209
            =0: RETURN;                                             07056
            >96: % Truncate string to avoid overflow. %             07057
               *tmpstr* _ SP, '", *astrng*[1 TO 96], '", SP;        07060
            ENDCASE                                                 07058
               *tmpstr* _ SP, '", *astrng*, '", SP;                 07059
         typeas($tmpstr);                                           06212
         RETURN;                                                    06214
         END;                                                       06215
      &llsrt _ namda.dalsrt;                                        03951
      olength _ dnstr.L;                                            03952
      *dnstr* _ NULL;                                               03953
      chrcnt _ 0;                                                   03954
      UNTIL (chrcnt _ chrcnt + 1) > astrng.L OR chrcnt > dnstr.M DO
                                                                    03965
         BEGIN                                                      03956
         CASE char _ *astrng*[chrcnt] OF                            03957
            =CA, =C., =LF, =CD, =BC, =BW, =$ascalt, IN [1B, 32B], IN
```

```
            [34B, 36B], =CR, =EOL: %an acceptable "non-printing"
            character%                                              03958
               BEGIN                                                03959
               &npstring _ npstrad(*astrng*[chrcnt]); %get
               representation of non-printing string%               03960
               IF npstring.L + dnstr.L > dnstr.M THEN EXIT LOOP;
                                                                    07128
               *dnstr* _ *dnstr*, *npstring*;                       03961
               END;                                                 03962
            ENDCASE *dnstr* _ *dnstr*, char;                        03963
         END;                                                       03964
      IF dnstr.L THEN                                               06196
         BEGIN                                                      06197
         *tmpstr* _ '", *dnstr*, '";                                06192
         IF olength THEN DSP( ...*tmpstr* )                         06200
         ELSE DSP( *tmpstr* );                                      06201
         END                                                        06198
      ELSE                                                          06199
         BEGIN                                                      06203
         cflstr.L _ MAX( 0, cflpos-1);                              06204
         cfldsp();                                                  06205
         END;                                                       06206
      RETURN;                                                       03969
      END.
                                                                    03970
%...subsystem name display...%                                      01420
   (dsubsys) PROCEDURE (astrng); %display in subsystem area%        03971
      %An A-string is displayed in  the subsystem area by calling
      dsubsys with the address of the A-string.%                    03972
      LOCAL llsrt;   REF llsrt, astrng;                             03974
      %--------------%                                             03973
      &llsrt _ subda.dalsrt;                                        03979
      llsrt.rtbps _ chbptr(empty) + &astrng;                        03980
      llsrt.rtbpe _ chbptr(astrng.L) + &astrng;                     03981
      wrtstr (&subda, &llsrt);                                      03982
      RETURN;                                                       03984
      END.
                                                                    03985
%...Command Feedback Line Routines...%                              01169
   %...CFL proper...%                                               01421
      (mrk1) PROCEDURE; %put CFL up arrow in first pos%             03986
         %----------------%                                        03987
         LOCAL ls;                                                  03988
         REF ls;                                                    03989
         &ls _ cflda.dalsrt + lsrtl;                                03990
         ls.rtx1 _ IF nldevice = devlproc THEN 0 ELSE 257;          03993
         %leave rtx2 as it was -- to overwrite old text%            04222
         arowon _ FALSE;                                            03995
         an();                                                      03996
         ls.rtx2 _ (cflarw.L-1)*ls.rthinc;                          03994
         RETURN;                                                    03997
         END.
                                                                    03998
      (mrk) PROCEDURE; %put CFL prompt in next position%            03999
         %----------------%                                        04000
         LOCAL  ls, llsrt, newpos, length;  REF ls, llsrt;          04001
```

```
   &llsrt _ (&ls _ cflda.dalsrt) + lsrtl;                          04002
   newpos _ (cflstr.L) * (ls.rthinc) + ls.rtx1;                    04251
      %position prompt string relative to CFL%                     04255
   llsrt.rtbpe _ chbptr( cflarw.L ) + $cflarw;                     04626
   IF newpos < (cflda.daright-cflda.daleft) THEN                   04627
      BEGIN                                                        04628
      IF nldevice = devlproc AND llsrt.rtx1 < newpos THEN          04003
         BEGIN %must erase begining of the old one%                04210
         length _ (newpos - llsrt.rtx1)/llsrt.rthinc;              04212
         cline (cflda.daleft+llsrt.rtx1, cflda.datop+llsrt.rty,
         length);                                                  04004
         END;                                                      04211
      llsrt.rtx1 _ newpos;                                         04005
      llsrt.rtx2 _ MAX(llsrt.rtx2, llsrt.rtx1 +
      (cflarw.L-1)*(llsrt.rthinc));                                04006
      wrtstr (&cflda, &llsrt);                                     04622
      llsrt.rtx2 _ llsrt.rtx1 + (cflarw.L-1)*(llsrt.rthinc);
                                                                   04213
      END;                                                         04629
   RETURN;                                                         04009
   END.
                                                                   04010
(mlf1) PROCEDURE (astrng); %move a-string to 1st pos of CFL%
                                                                   01186
   %Move Literal to Feedback line, first position.  This
   procedure sets the feeckback line position counter (CFLPOS)
   to 0.%                                                          01187
   %--------------------%                                         01188
   REF astrng;                                                     01189
   cflpos _ 0;                                                     01190
   *cflstr* _ *astrng*;                                            01191
   RETURN;                                                         01192
   END.
                                                                   01193
(mlf) PROCEDURE (astrng); %Move Literal to Feedback line%          01194
   %--------------------%                                         01195
   REF astrng;                                                     01196
   cflpos _ cflstr.L + 1;                                          01197
   *cflstr* _ *cflstr*, SP, *astrng*;                              01198
   RETURN;                                                         01199
   END.
                                                                   01200
(mlfr) PROCEDURE (astrng); %replace last literal in CFL%           01201
   %--------------------%                                         01202
   REF astrng;                                                     01203
   *cflstr* _ *cflstr* [1 TO cflpos], *astrng*;                    01204
   RETURN;                                                         01205
   END.
                                                                   01206
(cfldsp) PROCEDURE; %display CFLSTR in CFL area%                   04011
   %--------------------%                                         04012
   LOCAL ls, max, i, savex1;                                       04013
   REF ls;                                                         04014
   IF *savcfl* = *cflstr* THEN RETURN;                             04015
   &ls _ cflda.dalsrt;                                             04016
   max _ MAX(savcfl.L, cflstr.L);                                  04017
```

```
        ls.rtx2 _ MAX( ls.rtx1, max + ls.rtx1 - ls.rthinc);     05766
        i _ 1;                                                  05765
        IF nldevice = devlproc THEN                             05767
           FOR i UP UNTIL > max DO                              05763
              IF *cflstr*[i] NOT= *savcfl*[i] THEN EXIT LOOP;   05764
        savex1 _ ls.rtx1;                                       05893
        ls.rtx1 _ ls.rtx1 + (i-1)*ls.rthinc;                    05895
        *savcfl* _ *cflstr*;                                    04018
        ls.rtbps _ chbptr(i-1) + $cflstr;                       04019
        ls.rtbpe _ chbptr(cflstr.L) + $cflstr;                  04020
        wrtstr (&cflda, &ls );                                  04021
        ls.rtx1 _ savex1;                                       05894
        ls.rtx2 _ MAX( ls.rtx1, cflstr.L + ls.rtx1 - ls.rthinc);
                                                                04254
        RETURN;                                                 04022
        END.
                                                                04023
    %...CFL arrow...%                                           04372
       (an) PROCEDURE; %Turn the command feedback arrow on%     04373
        LOCAL ls;                                               04374
        REF ls;                                                 04375
        IF NOT (arowon := TRUE) THEN                            04376
           BEGIN                                                04377
           * cflarw * [2] _ '^;                                 04378
           &ls _ cflda.dalsrt + lsrtl;                          04379
           wrtstr (&cflda, &ls);                                04380
           END;                                                 04381
        RETURN;                                                 04382
        END.
                                                                04383
       (af) PROCEDURE; %Turn the command feedback arrow off%    04384
        LOCAL ls;                                               04385
        REF ls;                                                 04386
        IF (arowon := FALSE) THEN                               04387
           BEGIN                                                04388
           * cflarw * [2] _ '>;                                 04389
           &ls _ cflda.dalsrt + lsrtl;                          04390
           wrtstr (&cflda, &ls);                                04391
           END;                                                 04392
        RETURN;                                                 04393
        END.
                                                                04394
    %...CFL Question Mark...%                                   04395
       (qm)  PROCEDURE;                                         04396
        LOCAL ls;                                               04397
        REF ls;                                                 04398
        IF NOT (qmrkon := TRUE) THEN                            04399
           BEGIN                                                04400
           %display a '? below the command feedback line.%      04401
           * cflarw * [1] _ '?;                                 04402
           &ls _ cflda.dalsrt + lsrtl;                          04403
           wrtstr (&cflda, &ls);                                04404
           END;                                                 04405
        RETURN;                                                 04406
        END.
                                                                04407
```

```
      (qmoff)  PROCEDURE; %turn the "? off.%                          04408
         LOCAL ls;                                                    04409
         REF ls;                                                      04410
         IF (qmrkon := FALSE) THEN                                    04411
            BEGIN                                                     04412
            * cflarw * [1] _ SP;                                      04413
            &ls _ cflda.dalsrt + lsrtl;                               04414
            wrtstr (&cflda, &ls);                                     04415
            END;                                                      04416
         RETURN;                                                      04417
         END.
                                                                      04418
%...Literal feedback area...%                                         01653
   (rstlit) PROCEDURE;  %reset the literal display area%             010377
      LOCAL end, da, ls, endls, savelitline;                         010378
      LOCAL STRING send[20];                                         010379
      REF da, ls;                                                    010380
      IF nlmode # fulldisplay THEN RETURN;                           010381
      IF litreset THEN RETURN;                                       010382
      %clear the literal feedback area %                             010383
         savelitline _ litline;                                      010384
         clearlit();                                                 010385
      CASE nldevice OF                                               010386
         = devlproc:                                                 010387
            BEGIN                                                    010388
            %take care of tty windows!!%                             010389
               IF littyflag := FALSE THEN %restore normal tty window%
                                                                     010390
                  BEGIN                                              010391
                  lpttywindow(msgda.datop, msgda.dabottom);          010392
                  nlmode _ fulldisplay;                              010393
                  litapflag _ TRUE;                                  010394
                  savelitline _ litda.dabottom-litda.datop-1;        010395
                  END;                                               010396
            %restore suppressed file text display if LITAPFLAG set --
            otherwise let DAUPDATE or DAFRMT do it (with an
            appropriate initialization of the dalsupp fileds!)%
                                                                     010397
               lplitreset _ TRUE;                                    010398
               lplitline _ savelitline;                              010399
               IF (litapflag := FALSE) THEN %must restore old text%
                                                                     010400
                  BEGIN                                              010401
                  lplitreset _ FALSE;                                010402
                  end _ $dpyarea + dal*dacnt;                        010403
                  FOR &da _ $dpyarea UP dal UNTIL >= end DO          010404
                     IF  da.daexis AND NOT da.daseq THEN             010405
                        CASE TRUE OF %restore any strings that are
                        suppressed%                                  010406
                           = (da.dasuppress > 0): %restore display
                           area if it is suppressed%                 010407
                              resda(&da);                            010408
                           = (savelitline > 0): %restore suppressed
                           lines in a display area (if da.dalssup >
                           0)%                                       010409
                              BEGIN                                  010410
```

```
                        litreset _ TRUE;                    010411
                        dpset(dspjpf, endfil, endfil, endfil);
                                                            010412
                        daupdate(&da);  % the da.dalssup is set
                        to 0 in daupdate %                  010413
                        END;                                010414
                     ENDCASE  %no resetting has been done%
                                                            010415
                        lplitreset _ TRUE;                  010416
                  END;                                      010417
            track();                                        010418
            END;                                            010419
         = imlac0, = imlac1:                                010420
            BEGIN                                           010421
            IF littyflag := FALSE THEN                      010422
               BEGIN %should change to send char string also%  010423
               ttywindow(ttyda);                            010424
               nlmode _ fulldisplay;                        010425
               END;                                         010426
            %restore suppressed file text display%          010427
               end _ $dpyarea + dal*dacnt;                  010428
               FOR &da _ $dpyarea UP dal UNTIL >= end DO    010429
                  IF  da.daexis AND NOT da.daseq THEN       010430
                     BEGIN %restore any strings that are suppressed%
                                                            010431
                     %restore display area if it is suppressed%
                                                            010432
                        IF da.dasuppress THEN resda(&da);   010433
                     &ls _ da.dalsrt;                       010434
                     endls _ &ls + (da.dalssup := 0) * lsrtl;  010435
                        %dalssup points to the first string which is
                        not suppressed%                     010436
                     UNTIL &ls >= endls OR NOT ls.rtexis DO  010437
                        BEGIN                               010438
                        *send* _ begmsg, 4+remfudge, remrsda,
                        da.dahandle.daidr1 + remfudge,
                        da.dahandle.daidr2 + remfudge, ls.rtlsid;
                                                            010439
                        !sout(dspjfn, chbmty+$send, -send.L);  010440
                        IF da.dalhandle THEN %this display linked to
                        another -- restore it's string also%  010441
                           !sout(dspjfn, chbmty+$send, -send.L);
                                                            010442
                        &ls _ &ls + lsrtl;                  010443
                        END;                                010444
                     END;                                   010445
            litapflag _ FALSE;                              010446
            END;                                            010447
         ENDCASE                                            010448
            err($"No Tasker");                              010449
      litreset _ TRUE;                                      010450
      RETURN;                                               010451
      END.                                                  010452
                                                            010453
   (clearlit) PROCEDURE;  %Clear literal area%               04144
      %--------------%                                      04145
```

```
   LOCAL i;                                                          04146
   CASE nldevice OF                                                  04147
      = devlproc:                                                    04148
         BEGIN                                                       04149
         FOR i _ litda.datop UP litda.davinc UNTIL > litline
         +litda.datop DO                                           04150
            cline(0,i,lpxmax);                                     04151
         END;                                                        04152
      ENDCASE                                                        04153
         BEGIN                                                       04154
         clrseqda(litdahandle);                                    04155
            %clear literal area%                                   04156
         IF linkcnsl THEN clrseqda(litda.dalhandle);               04157
            %clear the literal feedback area for linked console%
                                                                    04158
         END;                                                        04159
   litline _ 0; litcolumn[litline] _ spacecol _ 1;                   04264
   RETURN;                                                           04160
   END.


                                                                     04161
(clrseqda) PROCEDURE(daid);  %clear sequential display area%         08498
   LOCAL save, da;                                                   08499
   LOCAL STRING send[10];                                            08500
   REF da;                                                           08501
   CASE nldevice OF                                                  08502
      = imlac0, = imlac1:                                            08503
         BEGIN                                                       08504
         *send* _ begmsg, 4+remfudge, remsdda, daid.daidr1 +
         remfudge, daid.daidr2 + remfudge, TRUE;                   08505
         !sout(dspjfn, chbmty+$send, -send.L);                     08506
         *send* _ begmsg, 3+remfudge, remrdda, daid.daidr1 +
         remfudge, daid.daidr2 + remfudge;                         08507
         !sout(dspjfn, chbmty+$send, -send.L);                     08508
         END;                                                        08509
      ENDCASE                                                        08510
         err($"No Tasker");                                         08511
   RETURN(daid);                                                     08512
   END.                                                              08513
                                                                     08514
(setlit) PROCEDURE;  %set up for literal collection%                 04095
   IF NOT litreset AND NOT litapflag THEN rstlit();                  04096
      %don't reset if in append mode%                                04097
   tabstore _ 4407B8 .V $littabs+1;                                  04098
   litreset _ FALSE;                                                 04099
   IF NOT litapflag THEN                                             04100
      BEGIN                                                          04101
      litline _ 0;                                                   04102
      litcolumn[litline] _ spacecol _ 1;                             04103
      END;                                                           04104
   RETURN;                                                           04107
   END.                                                              04108
                                                                     04109
(aplit) PROCEDURE (string); %append contents of string to literal
display area, removing file text as needed.  Takes care of tabs,
etc.%                                                                03028
```

```
   %-------------------%                                              03029
   LOCAL i;                                                          03381
   REF string;                                                       03031
   IF nlmode = typewriter THEN typeas(&string)                       03407
   ELSE                                                              03408
      BEGIN                                                          03409
      IF litreset THEN setlit();  %initialize lit machinery if
      first call%                                                    06222
      litapflag _ TRUE;  %set literal-area-in-append-mode flag%
                                                                     03438
      litstring _ &string;                                           03390
      litstore _ 440788 .V &string + 1;                              03391
         %byte pointer for literal feedback machinery%               03393
      FOR i _ string.L DOWN UNTIL <= 0 DO                            03392
         BEGIN                                                       06220
         IF littop + (litline+1)*litvinc >= litbottom THEN           07038
            BEGIN %lit area filled, ask user before going on%
                                                                     07039
            dismes(1,$"Type <OK> to see more, <CD> to stop");
                                                                     07040
            clrbuf(0);                                               07041
            input();                                                 07047
            dismes(0);                                               07046
            CASE curchr OF                                           07042
               = CD:                                                 07043
                  BEGIN                                              07052
                  rstlit();                                          07048
                  SIGNAL(statesig);                                  07049
                  END;                                               07051
               ENDCASE;                                              07044
            clearlit();                                              07050
            END;                                                     07045
         litchr(^litstore);                                          06218
         IF inptrf THEN EXIT LOOP;                                   06221
         END;                                                        06219
      END;                                                           03410
   RETURN;                                                           03402
   END.
                                                                     03048
(litdpy) PROCEDURE(astrng);  %Clear file text display windows and
display contents of string in literal area%                          04110
   %--------------%                                                  04111
   REF astrng;                                                       04112
   IF  nlmode = typewriter THEN typeas(&astrng)                      04113
   ELSE                                                              04114
      BEGIN                                                          04115
      supda(0); %suppress display of all text areas%                 04116
      clearlit();                                                    04117
      setlit(); %make litda area the tty-sim area%                   04118
      aplit(&astrng);                                                04119
      END;                                                           04120
   RETURN;                                                           04121
   END.

                                                                     04122
(littty) PROCEDURE(astrng);  %Clear file text display windows and
```

```
make lit area a tty window%                                          08556
   %--------------%                                                  08557
   LOCAL STRING send[10];                                            08558
   REF astrng;                                                       08559
   clearlit();                                                       08560
   supda(0); %suppress display of all text areas%                    08561
   setlit(); %set up lit area%                                       08562
   CASE nldevice OF                                                  08563
      = devlproc:                                                    08564
         BEGIN                                                       08565
         lpttywindow(litda.datop, litda.dabottom - litda.davinc);
                                                                     08566
         track();                                                    08567
         END;                                                        08568
      = imlac0, = imlac1:                                            08569
         BEGIN                                                       08570
         *send* _ begmsg, 4+remfudge, echda, 30B,
         litdahandle.daidr1 + remfudge, litdahandle.daidr2 +
         remfudge;                                                   08571
         !sout(dspjfn, chbmty+$send, -send.L);                       08572
         END;                                                        08573
      ENDCASE                                                        08574
         err($"No Tasker");                                          08575
   littyflag _ TRUE;                                                 08576
   nlmode _ typewriter;%should find some other way to do this%
                                                                     08577
   RETURN;                                                           08578
   END.

                                                                     08579
(linebreak) PROCEDURE;  %break the last line in the literal
feedback area%                                                       03152
   LOCAL i, count;                                                   03384
   IF spacecol IN (1, litcolumn[litline]) THEN                       03154
      BEGIN                                                          03155
      count _ MIN([litstring].L, litcolumn[litline] - spacecol);
                                                                     03397
      FOR i _ count DOWN UNTIL <= 0 DO litpbout(BC);                 03156
      litcolumn[litline] _ litcolumn[litline] - count;               04241
      END                                                            03158
   ELSE count _ 0;                                                   03159
   litpbout(EOL);                                                    03160
   litline _ MIN(litline + 1, 34);                                   03161
   litsup(littop + (litline+1)*litvinc);                             03162
   IF count > 0 THEN                                                 03163
      BEGIN                                                          03164
      %back litstore up count characters%                            03442
         FOR i _ count DOWN UNTIL <= 0 DO                            03443
            BEGIN                                                    03454
            %back up litstore%                                       03444
               r1 _ litstore;                                        03445
               r2 _ litstore-1;                                      03446
               UNTIL r2 = r1 DO                                      03447
                  BEGIN                                              03448
                  r3 _ r2;                                           03449
                  !IBP r2;                                           03450
```

```
                    END;                                                03451
                  litstore _ r3;                                        03452
               END;                                                     03453
         litcolumn[litline] _ 1;                                        04271
         IF littop + (litline+1)*litvinc >= litbottom THEN              07113
            BEGIN %lit area filled, ask user before going on%           07114
            dismes(1,$"Type <OK> to see more, <CD> to stop");           07115
            clrbuf(0);                                                  07116
            input();                                                    07117
            dismes(0);                                                  07118
            CASE curchr OF                                              07119
               = CD:                                                    07120
                  BEGIN                                                 07121
                  rstlit();                                             07122
                  SIGNAL(statesig);                                     07123
                  END;                                                  07124
               ENDCASE;                                                 07125
            clearlit();                                                 07126
            END;                                                        07127
         FOR i _ count DOWN UNTIL <= 0 DO litpbout(^litstore);          03441
         END;                                                           03167
      litcolumn[litline] _ count + 1;                                   03168
      spacecol _ 1;                                                     03169
      RETURN;                                                           03201
      END.                                                              03202
                                                                        03396
   (litsup) PROCEDURE (ycoord);                                         08515
      %suppress display of any line segments above ycoord on the
      screen%                                                           08516
      %-------------%                                                   08517
      LOCAL end, da, ls, stop, dtop;                                    08518
      LOCAL STRING send[20];                                            08519
      REF da, ls;                                                       08520
      end _ $dpyarea + dal*dacnt;                                       08521
      FOR &da _ $dpyarea UP dal UNTIL >= end DO                         08522
         IF da.daexis AND NOT da.daseg AND NOT da.dasuppress THEN
                                                                        08523
            IF (dtop _ da.datop) <= ycoord THEN                         08524
               BEGIN                                                    08525
               &ls _ da.dalsrt + da.dalssup * lsrtl;                    08526
               stop _  da.dalsrt + da.dalsz * lsrtl;                    08527
               UNTIL &ls >= stop DO                                     08528
                  BEGIN                                                 08529
                  IF ls.rtexis AND (dtop + ls.rty) <= ycoord THEN
                                                                        08530
                     BEGIN                                              08531
                     CASE nldevice OF                                   08532
                        = devlproc:                                     08533
                           BEGIN                                        08534
                           cline(da.daleft, ls.rty+dtop,
                           da.daright-da.daleft);                       08535
                           END;                                         08536
                        = imlac0, = imlac1:                             08537
                           BEGIN                                        08538
                           *send* _ begmsg, 5+remfudge, remssda,
                           da.dahandle.daidr1 + remfudge,
```

```
                        da.dahandle.daidr2 + remfudge,
                        ls.rtlsid, 0;                                   08539
                        !sout(dspjfn, chbmty+$send, -send.L);
                                                                        08540
                        IF da.dalhandle THEN                            08541
                           BEGIN                                        08542
                           !sout(ldspjfn, chbmty+$send, -send.L);
                                                                        08543
                           END;                                         08544
                        END;                                            08545
                     ENDCASE                                            08546
                        err($"No Tasker");                              08547
                  END                                                   08548
               ELSE EXIT;                                               08549
               &ls _ &ls + lsrtl;                                       08550
               END;                                                     08551
            da.dalssup _ (&ls-da.dalsrt)/lsrtl;                         08552
            END;                                                        08553
      RETURN;                                                           08554
      END.

                                                                        08555
%...Lesser support routines...%                                         01411
   (cntrlgetchar) PROCEDURE; %GETCHAR with CONTROL FILE STUFF%          08084
      %*****************************************************
      DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
      *****************************************************%           08085
      % This routine is just like GETCHAR except that it is used
      whenever the CONTROL FILE RECORD or PLAYBACK is in effect.%       08086
                                                                        08087
      %NOTE:  If inpwatchjfn is non-zero and recrunfil is zero then the
      time of char request, time when got char, and the character are
      written to the file for which inpwatchjfn is the jfn.  If
      inpwatchjfn is non-zero and recrunfil is also non-zero then only
      the character are written to the file for which inpwatchjfn is the
      jfn.  If inpjfn is not the controlling tty jfn then a simulated
      character wait is accomplished by means of the SIMCH jsys (this
      wait is the same as the original wait).  The format of this file
      is assumed to be the same as that built by this routine for
      non-zero inpwatchjfn.%                                            08088
      %-------------------%                                            08089
      LOCAL char, gottime, begtime;                                     08090
      IF inpjfn AND [rubmrk] %inptrf% THEN                              010641
         BEGIN  %playback, and ctrl-O was hit since last char%          010642
         ctlquit();                                                     010643
         RETURN(CD);                                                    010644
         END;                                                           010645
      %rubabt _ TRUE; allow command to be aborted%                      08091
      IF inpjfn THEN                                                    08092
         BEGIN %read time(request)%                                     08093
         r1 _ inpjfn;                                                   08094
         !JSYS bin;                                                     08095
         begtime.tmchr1 _ r2;                                           08096
         !JSYS bin;                                                     08097
         begtime.tmchr2 _ r2;                                           08098
         !JSYS bin;                                                     08099
```

```
      begtime.tmchr3 _ r2;                                              08100
      !JSYS bin;                                                        08101
      begtime.tmchr4 _ r2;                                              08102
      !JSYS bin;                                                        08103
      begtime.tmchr5 _ r2;                                              08104
      !JSYS bin;                                                        08105
      begtime.tmchr6 _ r2;                                              08106
      END                                                               08107
   ELSE                                                                 08108
      BEGIN                                                             08109
      !JSYS time;                                                       08110
      begtime _ r1;                                                     08111
      END;                                                              08112
   IF inpwatchjfn AND (NOT recrunfil) THEN                              08113
      BEGIN %write out time(request)%                                   08114
      r1 _ inpwatchjfn;                                                 08115
      r2 _ begtime.tmchr1;                                              08116
      !JSYS bout;                                                       08117
      r2 _ begtime.tmchr2;                                              08118
      !JSYS bout;                                                       08119
      r2 _ begtime.tmchr3;                                              08120
      !JSYS bout;                                                       08121
      r2 _ begtime.tmchr4;                                              08122
      !JSYS bout;                                                       08123
      r2 _ begtime.tmchr5;                                              08124
      !JSYS bout;                                                       08125
      r2 _ begtime.tmchr6;                                              08126
      !JSYS bout;                                                       08127
      END;                                                              08128
   %for ARC TENEX 131, there used to be calls to nlcrms around the
   !following section, TRUE and FALSE respectively, E.G. %              010461
      %IF nlcron AND nlstyp IN [0,2] THEN nlcrms(TRUE);%                010462
   IF inpjfn THEN                                                       08130
      BEGIN %sleep for appropriate time%                                08131
      %read time(gotchar)%                                              08132
         r1 _ inpjfn;                                                   08133
         !JSYS bin;                                                     08134
         gottime.tmchr1 _ r2;                                           08135
         !JSYS bin;                                                     08136
         gottime.tmchr2 _ r2;                                           08137
         !JSYS bin;                                                     08138
         gottime.tmchr3 _ r2;                                           08139
         !JSYS bin;                                                     08140
         gottime.tmchr4 _ r2;                                           08141
         !JSYS bin;                                                     08142
         gottime.tmchr5 _ r2;                                           08143
         !JSYS bin;                                                     08144
         gottime.tmchr6 _ r2;                                           08145
      %dismis for simulated char wait (time(gotchar) - time(requet))%
                                                                        08146
         IF symtflg THEN                                                08147
            BEGIN                                                       08148
            r1 _ gottime - begtime;                                     08149
            IF tenex < 13200 THEN !JSYS simch ELSE !JSYS dismis;        08150
            END;                                                        08151
      END;                                                              08152
```

```
   %get next character%                                                  08153
      r1 _ IF inpjfn THEN inpjfn ELSE ctty;                              08154
      !JSYS bin;                                                         08155
      char _ r2;                                                         08156
   %write out if being watched%                                          08158
      IF inpwatchjfn THEN                                                08159
         BEGIN                                                           08160
         %write time(gotchar)%                                           08161
            IF NOT inpjfn THEN                                           08162
               BEGIN                                                     08163
               !JSYS time;                                               08164
               gottime _ r1;                                             08165
               END;                                                      08166
            r1 _ inpwatchjfn;                                            08167
            IF NOT recrunfil THEN                                        08168
               BEGIN                                                     08169
               r2 _ gottime.tmchr1;                                      08170
               !JSYS bout;                                               08171
               r2 _ gottime.tmchr2;                                      08172
               !JSYS bout;                                               08173
               r2 _ gottime.tmchr3;                                      08174
               !JSYS bout;                                               08175
               r2 _ gottime.tmchr4;                                      08176
               !JSYS bout;                                               08177
               r2 _ gottime.tmchr5;                                      08178
               !JSYS bout;                                               08179
               r2 _ gottime.tmchr6;                                      08180
               !JSYS bout;                                               08181
               END;                                                      08182
         %write char%                                                    08183
            r2 _ char;                                                   08184
            !JSYS bout;                                                  08185
         END;                                                            08186
   inptrf _ inpstp _ rubabt _ FALSE;                                     08187
   RETURN(char);                                                         08188
   END.                                                                  08189

(initbtbl) PROCEDURE;  % initializes the global array SYSBREAKTBL to
contain appropriate addresses for RDLIT.  This routine should be
called after modifying the input translation table TRNSLI.%              01831
   LOCAL i;                                                              01890
   FOR i _ 0 UP UNTIL > 127 DO                                           01891
      CASE trnsli[i] OF                                                  04724
         = cachar,                                                       04725
         = rptchar,                                                      04726
         = optchar,                                                      04727
         = inschar: sysbreaktbl[i] _ $rdlcacdot;                         04728
         = CD: sysbreaktbl[i] _ $gps;                                    04729
         = BC: sysbreaktbl[i] _ $rdlbc;                                  04730
         = BW: sysbreaktbl[i] _ $rdlbw;                                  04731
         = TAB: sysbreaktbl[i] _ $rdladdchar;                            04732
         = todlit: sysbreaktbl[i] _ $rdlle;                              04733
         = $ctlr: sysbreaktbl[i] _ $rdlretype;                           04734
         = $ascbst: sysbreaktbl[i] _ $rdlbs;                             04735
         % commented out till defined -- smokey                          04736
         = IGNORE: sysbreaktbl[i] _ $rdladdchar;                         04737
```

```
        = SC: sysbreaktbl[i] _ $rdladdchar;                          04738
        = SW: sysbreaktbl[i] _ $rdladdchar;                          04739
        %                                                            04740
        ENDCASE sysbreaktbl[i] _ $rdladdchar;                        04741
   RETURN;                                                           01899
   END.                                                              01900
(inpvsp) PROCEDURE (char); %process view specs%                     010464
   LOCAL da, save, frmt;                                             010465
   REF da;                                                           010466
   LOCAL STRING locstr[20];                                          010467
   IF inhelp THEN RETURN;                                            010468
   IF msjfn THEN  %monitoring commands%                              010469
      msrecord(mmousespecs, 0, 0);                                   010470
   CASE char OF                                                      010471
      = 'F :  %recreate the display using dafrmt%                    010472
        BEGIN                                                        010473
        &da _ lda();                                                 010474
        IF sysvspec.vsrlev THEN                                      010475
           sysvspec _ <SEQGEN, reslev>(sysvspec, getlev(da.dacsp));
                                                                     010476
        da.dapvs _ da.davspec _ cspvs _ sysvspec;                    010477
        da.dapvs2 _ da.davspc2 _ cspvs[1] _ sysvspec[1];             010478
        save _ da.davspec.vsdaft := TRUE;                            010479
        dafrmt (&da, 0);                                             010480
        da.davspec.vsdaft _ save;                                    010481
        IF NOT save THEN dspvsp(da.davspec, da.davspc2, 3);          010482
        END                                                          010483
      = 'f :  %recreate the display using daupdate%                  010484
        BEGIN                                                        010485
        &da _ lda();                                                 010486
        IF sysvspec.vsrlev THEN                                      010487
           sysvspec _ <SEQGEN, reslev>(sysvspec, getlev(da.dacsp));
                                                                     010488
        da.davspec _ cspvs _ sysvspec;                               010489
        da.davspc2 _ cspvs[1] _ sysvspec[1];                         010490
        save _ da.davspec.vsdaft := TRUE;                            010491
        dpset(dspjpf, endfil, endfil, endfil);                       010492
        IF howformat(&da, da.dacsp.stfile, endfil : frmt) THEN
                                                                     010493
           IF frmt THEN                                              010494
              dafrmt(&da, 0)                                         010495
           ELSE                                                      010496
              daupdate (&da);                                        010497
        da.davspec.vsdaft _ save;                                    010498
        IF NOT save THEN dspvsp(da.davspec, da.davspc2, 3);          010499
        da.dapvs _ da.davspec;                                       010500
        da.dapvs2 _ da.davspc2;                                      010501
        END                                                          010502
      ENDCASE                                                        010503
        BEGIN                                                        010504
        sysvspec _ stklt(char, sysvspec, sysvspec[1] : sysvspec[1]);
                                                                     010505
           %call stklt to set viewspec%                              010506
        dn($vspstr);                                                 010507
        dspvsp(sysvspec, sysvspec[1], 3);                            010508
        END;                                                         010509
```

```
   RETURN;                                                          010510
   END.
                                                                    010511
(lkmkr) PROCEDURE (astrng, fileno); %translate marker name to t-ptr%
                                                                    0266
   %lkmkr converts a character pointer name (up to five characters)
   from the passed astring to a T-pointer for that marker.  It
   returns (ENDFIL) if that pointer does not exist.%                0267
   %----------------------%                                         0268
   LOCAL tmpnam, count, char, end, marker, flhd, stid;              0269
   REF astrng, marker;                                              0270
   stid _ origin;                                                   0271
   flhd _ filehead[stid.stfile _ fileno] - $filhed;                 0272
   &marker _ flhd + $mkrtb;                                         0273
   end _ &marker + [flhd + $mkrtbl]*mkrl;                           0274
   IF [flhd + $mkrtbl] <= 0 OR astrng.L <= empty THEN               0275
      RETURN(endfil);                                               0276
   tmpnam _ 0;                                                      0277
   count _ 1;                                                       0278
   CCPOS SF(*astrng*);                                              0279
   LOOP %extract marker tmpnam from a-string%                       0280
      BEGIN                                                         0281
      IF (char _ READC) = ENDCHR THEN EXIT;                         0282
      CASE count OF                                                 0283
         = 1: tmpnam.chr0 _ char;                                   0284
         = 2: tmpnam.chr1 _ char;                                   0285
         = 3: tmpnam.chr2 _ char;                                   0286
         = 4: tmpnam.chr3 _ char;                                   0287
         = 5: tmpnam.chr4 _ char;                                   0288
         ENDCASE EXIT;                                              0289
      BUMP count;                                                   0290
      END;                                                          0291
   DO IF marker.mkname = tmpnam THEN                                0292
      BEGIN   %depends on the extra bit being cleared by any routine
      adding markers to the marker table%                           0293
      stid.stpsid _ marker.mkpsid;                                  0294
      RETURN (stid, marker.mkccnt)                                  0295
      END                                                           0296
   UNTIL (&marker _ &marker + mkrl) = end;                          0297
   RETURN(endfil); %no such marker%                                 0298
   END.
                                                                    0299
(unput) PROCEDURE; % undo the effect of the last call to input %  044
   IF (buffs _ buffs - 1) < 0 THEN                                   053
      buffs _ buffsz;                                                054
   buff[ buffs ] _ curchr;                                           055
      % add last char read to front of input buffer%               01805
   curchr _ buff[ IF buffs = 0 THEN buffsz ELSE buffs - 1 ];         059
   RETURN;                                                           060
   END.                                                              061
(resetb) % resets the character input buffer %                    01787
   % resetb is called with one argument: the address of a text string
                                                                    01788
    this string is placed at the beginning of the character input
   buffer so that subsequent calls to input() will return the
   character string provided.                                       01789
```

```
   This routine provides a mechanism for returning already processed
   characters to the input buffer to be reread as new input, i.e. it
   can be used to back up the input stream. %                    01790
   PROCEDURE( astrng );                                          01791
      LOCAL char;                                                01792
      REF astrng;                                                01793
      IF astrng.L = 0 THEN RETURN;                               01878
      CCPOS SE(*astrng*);                                        01879
      LOOP CASE char _ READC OF                                  01880
         = ENDCHR:                                               01881
            BEGIN                                                01886
            curchr _ buff[ buffs ];                              01802
            RETURN;                                              01803
            END;                                                 01887
         ENDCASE                                                 01882
            BEGIN                                                01883
            IF (buffs _ buffs - 1) < 0 THEN                      01796
               buffs _ buffsz;                                   01797
            buff[ buffs ] _ char; % add char to input buffer%    01798
            IF buffn = buffs % check for buffer full %           01799
               THEN err( $"Input Buffer Overflowed" );           01800
            END;                                                 01884
      END.                                                       01804
(cvsno) PROCEDURE(astr); %convert a-string to number%             0973
   %An A-string containing digits can be converted to a binary
   integer by calling CVSNO with the A-string address.  Theconversion
   is done to the base 10, and all characters are assumed to be
   digits, and are not checked.  The A-string is not changed, and the
   integer is returned.  If the first character is a  minus sign, the
   number will be converted to a negative number correctly.%     0974
   %-----------------%                                          0975
   LOCAL                                                        0976
      chrcnt, %character count for char readout%                0977
      negnum, %negative number flag%                            0978
      number; %cell in which the number is constructed%         0979
   REF astr;                                                    0980
   number _ negnum _ 0;                                         0981
   chrcnt _ empty + 1;                                          0982
   IF *astr*[chrcnt] = '- THEN BUMP chrcnt, negnum;             0983
   DO number _ number*10 + (*astr*[chrcnt] - '0)                0984
   UNTIL (chrcnt := chrcnt + 1) = astr.L;                       0985
   RETURN(IF negnum THEN -number ELSE number);                  0986
   END.
                                                                0987
(specttyout)PROC(ttyno, astr);                                 08210
   %This procedure accepts a teletype numbr and a string, and types
   the string as a message on the tty, or to jlog file if ttyno=0.
                                                                08211
      It additionally typeas a series of attention getting characters
      on the tty.%                                              08212
   LOCAL STRING error[200];                                     08213
   LOCAL ttjfn, ttlocjfn;                                       08214
   REF astr;                                                    08215
   ttlocjfn _ 0;                                                08216
   IF jdebug  OR ttyno<0 THEN RETURN; %Don't do it if debugging or if
   passed neg. ttyno%                                           08217
```

```
   IF ttyno=0 THEN                                                    08218
      BEGIN                                                           08219
      IF jlogjfn=0 THEN                                               08220
         BEGIN                                                        08221
         IF NOT ttlocjfn _ sgtjfn (0, $"<JOURNAL>JLOG.TXT;P777724",
         $error) THEN                                                 08222
            BEGIN                                                     08223
            *error* _ "Can't get JFN for JLOG file:  ", *error*;
                                                                      08224
            specttyout (oprtty,$error);                               08225
            RETURN;                                                   08226
            END;                                                      08227
         IF NOT sysopen (ttlocjfn, append, chrtyp, $error) THEN       08228
            BEGIN                                                     08229
            IF NOT SKIP !rljfn(ttlocjfn) THEN NULL;                   08230
            *error* _ "Trying to log journal error:  ", *error*;
                                                                      08231
            specttyout (oprtty,$error);                               08232
            RETURN;                                                   08233
            END;                                                      08234
         ttjfn_ttlocjfn;                                              08235
         END                                                          08236
      ELSE ttjfn_100B;                                                08237
      END                                                             08238
   ELSE                                                               08239
      BEGIN                                                           08240
      %Check that tty is available%                                   08241
         r1 _ getsbn($"TTYJOB");                                      08242
         !JSYS sysgt;                                                 08243
         IF (r1.RH _ r2.RH) = 0 THEN RETURN; %Table not thee%         08244
         r1.LH _ ttyno;                                               08245
         IF (NOT SKIP !JSYS getab) OR (r1 # -1) THEN RETURN;          08246
      %set ttjfn for tty%                                             08247
         ttjfn_ 405 .V ttyno;                                         08248
      END;                                                            08249
   %First the attention getters%                                      08250
      !sout(ttjfn, chbptr(0) + $"_____--", -7);                       08251
   %Now the string%                                                   08252
      !sout(ttjfn, chbptr(0) + &astr, -astr.L);                       08253
   %Now another EOL%                                                  08254
      !bout(ttjfn, CR); !bout(r1, LF);                                08255
   IF ttlocjfn AND NOT sysclose(ttlocjfn, $error) THEN dismes(1,
   $error);                                                           08256
   RETURN                                                             08257
   END.                                                               08258
%...tab support routines...%                                          02218
   (fndtab) PROCEDURE (da, column);  %find new tab position%          02219
      %This procedure returns the column position of the next         02220
      tab, given a column position in column%                         02221
      %-----------------%                                            02222
      LOCAL tabtbl[3];                                                05942
      REF da;                                                         02223
      tabtbl _ da.datab0; %setup local tab table (adress shit $&#")%
                                                                      05943
      tabtbl[1] _ da.datab1;                                          05944
      tabtbl[2] _ da.datab2;                                          05945
```

```
      RETURN(nxtbit($tabtbl, column, 3));                                05946
      END.                                                               02236
                                                                         02237
FINISH of inpfbk                                                         01261
%                                                                        01408
(rawchr)  %%Should be after FINISH statement!!!%%                        01407
   RAWCHR is no longer a routine but is the address of the lowest level
   character input routine.  Under normal conditions, it points to
   GETCHAR.  If we are in Control File Record or Playback mode or if
   MSFLAG is set, then it should point to cntrlgetchar.                  01410
%                                                                        01409
%this procedure used to do measurement for ARC TENEX 131 which had a
special JSYS%                                                           010463
   (nlcrms) PROCEDURE(ndofcom);  %Issue NLSCR JSYS%                      07722
      %This routine handles the calls to the NLSCR jsys from the input
      routine.  If the argument is TRUE then it assumes we are at the
      end of a "commnd" (character interaction), otherwise, at the
      beginning.%                                                        07723
      LOCAL curtim, extim;                                               07724
      IF tenex >= 13200 THEN RETURN;                                     07725
      r1 _ -5;                                                           07726
      !JSYS jobtm;                                                       07727
      extim _ r1; %result in millisecs%                                  07728
      !JSYS time;                                                        07729
      curtim _ r1; % result in milliseconds %                            07730
      IF ndofcom THEN                                                    07731
         BEGIN                                                           07732
         IF nlcrlst THEN %not first call to jsys%                        07733
            BEGIN                                                        07734
            r1 _ curtim - nlcrrt;                                        07735
            r2 _ extim - nlcret;                                         07736
            r3 _ curtim - (nlcrlst := curtim);                           07737
            r4 _ nlstyp;                                                 07738
            !JSYS nlscr;                                                 07739
            END                                                          07740
         ELSE nlcrlst _ curtim;                                          07741
         END                                                             07742
      ELSE                                                               07743
         BEGIN                                                           07744
         nlcret _ extim;                                                 07745
         nlcrrt _ curtim;                                                07746
         END;                                                            07747
      RETURN;                                                            07748
      END.
                                                                         07749
```

JNL DEL

```
< NLS, JNLDEL.NLS.39, >, 4-Dec-78 17:37 JCP ;;;;
FILE jnldel % L10 <rel-NLS>jnldel %% (L10,) (rel-nls,jnldel.rel,) %
%Declarations %
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4;
   REF tda, ojsqsw;
   REF rawchr;
%Declarations%
   DECLARE jmaxsmt =  5000;
   DECLARE sigdsf = 40215; %discfull signal%
   DECLARE STRING xlit[2000];
% Journal maintenance utility execution %
   (joutil)PROCEDURE(libf,idstr);
      % bits of libf as follows%
         % 1B: run recover files %
         % 2B: run on-line distribution %
         % 4B: run slinker %
         % 10B: dettach, go to sleep, on hour run on-line dist. and
         slinker %
         % 20B: Auto Hard Copy%
         % 40B: update initial files at on-line delivery%
         % 100B: ask operator for input files to process%
         % 200B: deliver only to idents found in idstr%
         % 400B: ask operator for document numbers to deliver%
         % liblod TRUE: don't set mlav from jdriver file (it was set by
         hand) %
      LOCAL curtime, libf1;
      LOCAL STRING error[100];
      % Initialize %
         %  Set up operators initials to be XXX %
            *initsr* _ "XXX";
            curtime _ 'X - 100B;
            cinit _ 0;
            cinit.cint1 _ curtime;
            cinit.cint2 _ curtime;
            cinit.cint3 _ curtime;
         % set master access print flag; determines whether hard copy
         file should be printed %
            setmastac();
         % get all of background's names in system table %
            !setnm(getsbn($"OLJDEL"));
            !setnm(getsbn($"OJHLAV"));
            !setnm(getsbn($"SLINKR"));
            !setnm(getsbn($"HCJDEL"));
            !setnm(getsbn($"SNKSLP"));
         % connect to JOURNAL directory %
            conjdir(TRUE);
      %Make sure system date and time have been set; we can't wake up if
      they aren't.  Type out nasty message every twenty minutes on the
      operator's teletype if it hasn't been set%
         timcheck();
      % Run recover files.  Check validity of Journal system files
      including content specified in JDRIVER file's RECOVERLIST branch%
         IF (libf .A 1B) THEN xrecovf();
      % If nothing else is to be done, then RETURN. %
         IF (libf _ libf .X 1) = 0 THEN RETURN;
```

```
% Turn off recovf bit so recover files will not be done again% 050
   libf _ libf .A -2;                                          051
IF autostrt THEN                                               052
   BEGIN                                                       053
   %Set controlling tty up as primary output again%            054
      r1 _ 4B5;                                                055
      !JSYS gpjfn;                                             056
      r2.RH _ -1;                                              057
      r1 _ 4B5;                                                058
      !JSYS spjfn;                                             059
      autostrt_FALSE;                                          060
      IF jlogjfn AND NOT sysclose(jlogjfn:=0, $error) THEN
      dismes(1, $error);                                       061
   END                                                         062
ELSE IF (libf .A 10B) THEN                                     063
   BEGIN                                                       064
   % Detach and continue processing. %                         065
   crlf();                                                     066
   typeas($"Detaching");                                       067
   !JSYS dtach;                                                068
   END;                                                        069
LOOP % Loop around, processing on instructions from operator the
first time, from the JDRIVER file on successive rounds when
awakened. %                                                    070
   BEGIN                                                       071
   % Set time table based on values in JDRIVER file; set load
   average cut off if it has not been set already. %           072
      jsettitab();                                             073
   % Distribute on-line documents if requested. %              074
      IF (libf .A 2B) THEN                                     075
         BEGIN                                                 076
         % Set up JCAT entry. %                                077
         % Build distribution file entry. %                    078
         %Distribute on-line documents%                        079
            oldist(libf,idstr);                                080
         END;                                                  081
   % Do hard copy distribution printing.  If bit 10B is on, it
   will be run detached with a primary output file set up. %   082
      IF libf .A 20B THEN  autohcd(libf .A 10B);               083
   % Verify and update important files listed in the SLINKERLIST
   branch of JDRIVER.  If things are not OK, lock the Journal
   system. %                                                   084
      IF libf .A 4B THEN  slinker();                           085
   % If we are to detach and wake up later (bit 10B on), set
   default flag mode bis if necessary, then go to sleep; otherwise
   RETURN. %                                                   086
      IF NOT (libf .A 10B) THEN RETURN;                        087
      libf_libf .V 16B; %default is on-line deliv and slinker% 088
      r1 _ getsbn($"SNKSLP");                                  089
      !JSYS setnm; %set up name for sleeping%                  090
      %Make sure system date and time have been set; we can't wake
      up if they aren't.  Type out nasty message every twenty
      minutes on the operator's teletype if it hasn't been set%
                                                               091
         timcheck();                                           092
      r2 _ -1;                                                 093
```

```
            r4 _ 0;                                                    094
            !JSYS odcnv;                                               095
            IF r4 = -1 THEN                                            096
               r1 _ 1800000 %system does not have time%                097
            ELSE                                                       098
               BEGIN                                                   099
               curtime _ (r4.RH)/60;                                   0100
               % getojtime looks in the time table which was set up from
               JDRIVER to find out when to wake up next.  It returns the
               time as well as the new value of libflg to be used the
               next time through the system. %                         0101
               IF (r1 _ getojtime(curtime + 10 : libf1)) # 0 THEN      0102
                  r1 _ r1 - curtime                                    0103
               ELSE r1 _ 24 * 60 - curtime + getojtime(0 : libf1);     0104
               r1 _ r1 * 60000; %Convert to ms%                        0105
               END;                                                    0106
         !JSYS dismS;                                                  0107
         IF libf1 THEN libf _ libf1;                                   0108
         END;                                                          0109
      RETURN;                                                          0110
      END.
                                                                       0111
% Verification of Journal system files. %                             0112
   (xrecovf) PROC;                                                     0113
      % "Recover" journal system files at start up and when requested.
      This procedure checks the validity of files in the RECOVERLIST
      branch of JDRIVER using te procedure CHECKFILE which checks not
      only the NLS validity through a file verify, but also checks for
      the existence of statement names and content as specified in the
      RECOVELIST statement.  It is a more thorough (and slow) process
      than SLINKER.  If anything is wrong, every attempt is made to
      create an unlocked, VALID system file.    The journal is locked
      while this process goes on.  It is unlocked only if everthing is
      OK%                                                              0114
      LOCAL fileno, stid, faterr, recstid;                             0115
      LOCAL TEXT POINTER z1, z2;                                       0116
      LOCAL STRING error[100];                                         0117
      faterr _ 0;                                                      0118
      ON SIGNAL                                                        0119
         =-5: %We Got here on a badfile, and who knows how%            0120
            IF fileno = bfilno THEN                                    0121
               BEGIN                                                   0122
               close(fileno := 0);                                     0123
               typeas($"File List Bad File");                          0124
               r1 _ 4B5; !JSYS haltf;                                  0125
               END                                                     0126
            ELSE                                                       0127
               BEGIN                                                   0128
               lockjo(1);                                              0129
               typeas($"Bad File--");                                  0130
               typeas($lit);                                           0131
               r1 _ 4B5; !JSYS haltf;                                  0132
               END;                                                    0133
         ELSE                                                          0134
            BEGIN                                                      0135
            crlf();                                                    0136
```

```
        typeas(sysmsg);                                                 0137
        crlf();                                                         0138
        typeas($"Fatal Error");                                         0139
        r1 _ 4B5;                                                       0140
        !JSYS haltf;                                                    0141
        END;                                                            0142
lockjo(0); %Lock Journal%                                               0143
%Detatch and Set up jlog file as primary output file%                   0144
   IF autostrt AND jlogjfn=0 THEN                                       0145
      BEGIN                                                             0146
      IF NOT jlogjfn _ sgtjfn (0, $"<JOURNAL>JLOG.TXT;P777724",
      $error) THEN                                                      0147
         BEGIN                                                          0148
         *error* _ "Can't get JFN for JLOG file:  ", *error*;           0149
         specttyout (oprtty,$error);                                    0150
         typeas($error);                                                0151
         LOOP !haltf();                                                 0152
         END;                                                           0153
      IF NOT sysopen (jlogjfn, append, chrtyp, $error) THEN             0154
         BEGIN                                                          0155
         IF NOT SKIP !rljfn(jlogjfn) THEN NULL;                         0156
         *error* _ "Trying to log journal error:  ", *error*;           0157
         specttyout (oprtty,$error);                                    0158
         typeas($error);                                                0159
         LOOP !haltf();                                                 0160
         END;                                                           0161
      !gpjfn(4B5);                                                      0162
      r2.RH _ jlogjfn;                                                  0163
      !spjfn(4B5);                                                      0164
      END;                                                              0165
r1 _ getsbn($"RECOVF");                                                 0166
!JSYS setnm;                                                            0167
conjdir(TRUE); %Connect to JOunal Directory%                            0168
enab!access(0, jrnlaccess);                                             0169
IF enablw() = -1 AND NOT nwheelf THEN                                   0170
   BEGIN                                                                0171
   crlf();                                                              0172
   typeas($"You need WHEEL capability to run recover files");           0173
   r1 _ 4B5;                                                            0174
   !JSYS haltf;                                                         0175
   END;                                                                 0176
stid _ 0;                                                               0177
stid.stpsid _ origin;                                                   0178
IF (stid.stfile _ fileno _ rawopen(jflname($"jdriver"), FALSE)) =
0 THEN                                                                  0179
   BEGIN                                                                0180
   typeas($"Driver File open fail");                                    0181
   !haltf(4B5);                                                         0182
   END;                                                                 0183
IF (recstid _ namelook(stid, $"recoverlist")) = endfil THEN snkerr
($"No recoverlist in driver file");                                     0184
stid _ getsub(recstid);                                                 0185
LOOP                                                                    0186
   BEGIN                                                                0187
   IF stid = recstid THEN EXIT;                                         0188
   CCPOS SF(stid);                                                      0189
```

```
      FIND $NP ^z1 ([NP] < CH > / SE(stid)) ^z2;                    0190
      *lit* _ z1 z2;                                                0191
      IF NOT checkfile(jflname($lit), $z2) THEN BUMP faterr;        0192
      stid _ getsuc(stid);                                          0193
      END;                                                          0194
   close(fileno:=0);                                                0195
   %Unlock Journal and proceed if no fatal errors%                  0196
      IF faterr THEN                                                0197
         BEGIN                                                      0198
         typeas($"Fatal File Errors--Journal Left Locked");         0199
         lockjo(1); %just to be sure%                               0200
         !JSYS haltf;                                               0201
         END;                                                       0202
      unlkjo(-1); %allow Journal usage%                             0203
   RETURN;                                                          0204
   END.
                                                                    0205
(slinker) PROC ;                                                    0206
   %Slink around, verify, and update journal files listed in
   JDRIVER's SLINKERLIST branch.  Expunge directory if requested.  If
   a file is bad, the fact is noted.  The files are verified twice if
   locked, once if not.  Unlocks journal system if everything is OK;
   locks it if error detected in one of the listed files; terminates
   through SNKERR if other error.%                                  0207
   %------------------%                                            0208
   LOCAL stid1, stid, busycnt, slistid;                             0209
   LOCAL TEXT POINTER z1, z2;                                       0210
   stid _ stid1 _ 0;                                                0211
   ON SIGNAL                                                        0212
      =-5: %We Got here on a badfile, and who knows how%            0213
         IF stid.stfile = bfilno THEN                               0214
            BEGIN                                                   0215
            close(stid.stfile  := 0);                               0216
            snkerr($"File List Bad File");                          0217
            END                                                     0218
         ELSE                                                       0219
            IF stid1.stfile =  bfilno THEN                          0220
               (lockfiles): BEGIN                                   0221
               lockjo(1);                                           0222
               *lit* _ "Bad File--Probably ", *jnamstr*, CR, LF,
               *lit*;                                               0223
               snkerr($lit);                                        0224
               END;                                                 0225
      =-6: %I/O data error%                                         0226
         BEGIN                                                      0227
         *lit* _ *[sysmsg]*;                                        0228
         GOTO lockfiles;                                            0229
         END;                                                       0230
      ELSE                                                          0231
         snkerr(sysmsg);                                            0232
   r1 _ oljsbn _ getsbn($"SLINKR");                                 0233
   !JSYS setnm;                                                     0234
   enablaccess(0, jrnlaccess);                                      0235
   stid _ 0;                                                        0236
   stid.stpsid _ origin;                                            0237
   stid.stfile _ open(0, jflname($"jdriver"));                      0238
```

```
   IF (slistid _ namelook(stid, $"slinkerlist")) = endfil THEN    0239
      snkerr ($"No slinkerlist in driver file");                  0240
   stid _ getsub(slistid);                                        0241
   LOOP                                                           0242
      BEGIN                                                       0243
      IF stid = slistid THEN EXIT;                                0244
      IF discfull(1400) THEN EXIT;                                0245
      CCPOS SF(stid);                                             0246
      FIND $NP ^z1 ([NP] / SE(stid)) ^z2;                         0247
      *lit* _ z1 z2;                                              0248
      busycnt _ 0;                                                0249
      stid1.stpsid _ origin;                                      0250
      LOOP                                                        0251
         BEGIN                                                    0252
         oplock(1);                                               0253
         IF (stid1.stfile _ rawopen(jflname($lit), 0)) THEN EXIT;
                                                                  0254
         oplock(0);                                               0255
         IF (busycnt _ busycnt+1) > 3 THEN                        0256
            BEGIN                                                 0257
            stid _ getsuc(stid);                                  0258
            REPEAT LOOP 2;                                        0259
            END;                                                  0260
         !dismS(15000);                                           0261
         END;                                                     0262
      oplock(0);                                                  0263
      crng(TRUE, stid1.stfile);                                   0264
      csdb(TRUE, stid1.stfile);                                   0265
      ckstrc(stid1);                                              0266
      IF updtfl(stid1.stfile, 1, 0) THEN                          0267
         BEGIN                                                    0268
         setaccess(stid1.stfile, jrnlaccess); %reset access%      0269
         crng(TRUE, stid1.stfile);                                0270
         csdb(TRUE, stid1.stfile);                                0271
         ckstrc(stid1);                                           0272
         END;                                                     0273
      close(stid1.stfile);                                        0274
      stid _ getsuc(stid);                                        0275
      END;                                                        0276
   close(slistid.stfile);                                         0277
   %Now expunge if flag set%                                      0278
      IF flagut(5, $tstfg) THEN                                   0279
         BEGIN                                                    0280
         !JSYS gjinf;                                             0281
         r1 _ r2; %connected directory number%                    0282
         !JSYS deldf; %expunge files%                             0283
         END;                                                     0284
   unlkjo(1); %alright to use it now%                             0285
   RETURN;                                                        0286
   END.
                                                                  0287
(checkfile) PROC (astr, z2);                                      0288
   %open the file named in astr, an do anything necessary to assure
   that it is left as an unlocked and legal NLS file%             0289
   %Return false if there was no trace of file, otherwise true%   0290
   %-----------------%                                            0291
```

```
   LOCAL fileno, retry, dirno, type, stid, jfn;                    0292
   LOCAL TEXT POINTER z1, z3;                                      0293
   LOCAL STRING lookst[50];                                        0294
   REF astr, z2;                                                   0295
   dismes(1, &astr);                                               0296
   retry _ -1;                                                     0297
   (chloop):                                                       0298
   BUMP retry;                                                     0299
   stid _ 0;                                                       0300
   stid.stpsid _ origin;                                           0301
   IF NOT stid.stfile _ fileno _ rawopen(&astr, TRUE) THEN RETURN(0);
                                                                   0302
   ON SIGNAL                                                       0303
      =-5: %Bad File%                                              0304
         BEGIN                                                     0305
         dismes(1, $"Bad File");                                   0306
         RETURN(0);                                                0307
         END;                                                      0308
      =-6: %I/O data error%                                        0309
         BEGIN                                                     0310
         dismes(1, $"I/O data error");                             0311
         RETURN(0);                                                0312
         END;                                                      0313
      ELSE;                                                        0314
   cverfil(fileno, 1); %verify file%                               0315
   %do update if there is enough (300 pgs) disc%                   0316
      !gdskc(600000777777B);                                       0317
      IF r2 > 300 THEN updtfl(fileno, 1, 0);                       0318
   WHILE NOT (FIND z2 $NP ENDCHR) DO %check for content or name%   0319
      BEGIN                                                        0320
      type _ 0;                                                    0321
      IF FIND z2 "Name: " $NP ^z1 L $LD ^z2 ^z3 THEN               0322
         type _ nametyp                                            0323
      ELSE                                                         0324
         IF FIND z2 "Content: " $NP ^z1 [';] ^z2 ^z3 _z3 THEN      0325
            type _ contnt                                          0326
         ELSE                                                      0327
            FIND z2  $NP [NP/ENDCHR] ^z2;                          0328
      IF type THEN                                                 0329
         BEGIN                                                     0330
         *lookst* _ z1 z3;                                         0331
         FIND SF(stid) ^z1;                                        0332
         lookup($z1, $lookst, type);                               0333
         IF z1 = endfil THEN                                       0334
            BEGIN                                                  0335
            dismes(1, $"Logical File Error");                      0336
            RETURN(0);                                             0337
            END;                                                   0338
         END;                                                      0339
      END;                                                         0340
   setaccess(fileno, jrnlaccess); %Reset access in case it has
   changed mysteriously%                                           0341
   close(fileno);                                                  0342
   RETURN(TRUE);                                                   0343
   END.
                                                                   0344
```

```
% Build distribution file and catalog entries %                          0345
  (jinsubs)PROC(flgs, idfilno); %experimental version: takes secondary
  distribution requests in tejournal%                                    0346
    %processes submitted items in <TEJOURNAL>%                           0347
    %Bit 1008: go to operator for input files to process%                0348
    LOCAL tstid, dirf, deferd, inftyp, nethcflag;                        0349
    LOCAL usednum, jcstid, jnlstid, jfn, mesflg, secdflag, recorded,
    sextloc;                                                             0350
    LOCAL TEXT POINTER z1, z2, z3, z4, z5;                               0351
    LOCAL STRING tempsr[100], filenm[50], mfname[50], clrkstr[15],
    remfname[30];                                                        0352
    LOCAL STRING number[5], modstr[100], dfname[50], dirname[30],
    dayext[10], headstr[80], sourcedir[30] %input%, infstra[36];         0353
    jcstid _ jnlstid _ 0;                                                0354
    ON SIGNAL ELSE                                                       0355
       BEGIN                                                             0356
       sigclose(diststid.stfile := 0);                                   0357
       sigclose(jcstid.stfile := 0);                                     0358
       sigclose(jnlstid.stfile := 0);                                    0359
       END;                                                              0360
    CASE jdebug OF                                                       0361
       = TRUE: *sourcedir* _ "<DUVALL>";                                 0362
       ENDCASE *sourcedir* _ "<TEJOURNAL>";                              0363
    %make up extension "NLn" where n is the last digit of the date%
                                                                         0364
       *datesr* _ NULL;                                                  0365
       dtfrmt(-1,$datesr);                                               0366
       FIND SF(*datesr*) [1$D] ^z2 < D ^z1;                              0367
       *dayext* _ "NL", z1 z2;                                           0368
    inftyp _ IF jolock() THEN 2 ELSE 3;                                  0369
    jfn _ 0;                                                             0370
    IF inremsiteflag AND NOT discfull(1300) THEN                         0371
       %bring across outjournal files from other machine if:%            0372
          %load average is low on both machines%                         0373
          %adequate file space on this machine%                          0374
       BEGIN                                                             0375
       IF lhostn = archost THEN                                          0376
          BEGIN                                                          0377
          ftprcop ($utilstr, $"OUTJOURNAL", $jnlpsw, $"0", $dayext,
          $"<TEJOURNAL>", $"NLR", ';);                                   0378
          END                                                            0379
       ELSE                                                              0380
          BEGIN                                                          0381
          ftprcop ($arcstr, $"OUTJOURNAL", $jnlpsw, $"0", $dayext,
          $"<TEJOURNAL>", $"NLR", ';);                                   0382
          END;                                                           0383
       END;                                                              0384
    %position distribution stid%                                         0385
       diststid _ getail(getsub(openlock(0, jflname($"distfile"))));
                                                                         0386
    LOOP BEGIN %process input files%                                     0387
       %do tejournal processing if 700 disc pages left%                  0388
          IF discfull(1700) THEN EXIT LOOP;                              0389
       %get an NLS file from TEJOURNAL%                                  0390
          IF flgs .A 100B THEN BEGIN %go to user for file name%          0391
             echon();                                                    0392
```

```
        crlf();                                                    0393
        typeas($"Input process ");                                 0394
        echoff();                                                  0395
        dirf _ FALSE;                                              0396
        CASE lookc() OF                                            0397
           ='f, ='F, =CA: BEGIN                                    0398
             .input();                                             0399
              IF lookc() = '< THEN dirf _ TRUE;                    0400
              END;                                                 0401
           ='q, ='Q: BEGIN                                         0402
              input();                                             0403
              typeas($"Quit");                                     0404
              %getctc();%                                          0405
              EXIT LOOP;                                           0406
              END;                                                 0407
           ='a, ='A: BEGIN                                         0408
              input();                                             0409
              typeas($"All Files");                                0410
              %getctc();%                                          0411
              flgs _ flgs .X 100B;                                 0412
              jfn _ 0;                                             0413
              REPEAT LOOP;                                         0414
              END;                                                 0415
           ='<: dirf _ TRUE;                                       0416
           ENDCASE;                                                0417
        typeas($" File ");                                         0418
        echon();                                                   0419
        *filenm* _ NULL;                                           0420
        IF rdlit($filenm,0) # 1 THEN BEGIN                         0421
           typeas ($"? ");                                         0422
           REPEAT LOOP;                                            0423
           END;                                                    0424
        IF NOT dirf THEN *filenm* _ *sourcedir*, *filenm*;         0425
        echoff();                                                  0426
        IF NOT (jfn_sgtjfn(100101B6,$filenm, $tempsr)) THEN
        REPEAT LOOP;                                               0427
        END                                                        0428
      ELSE BEGIN                                                   0429
        IF NOT jfn THEN CASE (inftyp _ inftyp - 1) OF              0430
           =2: BEGIN %defered%                                     0431
              deferd _ TRUE;                                       0432
              sextloc _ $"NLP";                                    0433
              END;                                                 0434
           =1: BEGIN %local numbered%                              0435
              deferd _ FALSE;                                      0436
              sextloc _ $"NLN";                                    0437
              END;                                                 0438
           =0: BEGIN %remote numbered%                             0439
              deferd _ FALSE;                                      0440
              sextloc _ $"NLR";                                    0441
              END;                                                 0442
           ENDCASE EXIT LOOP;                                      0443
        *infstra* _ *sourcedir*, "*.", *[sextloc]*, fvrdchar, '*;
                                                                   0444
        IF NOT (jfn_sgtjfn(100101B6, $infstra, $tempsr)) THEN
        REPEAT LOOP;                                               0445
```

```
      END;                                                          0446
   % rename  with "NLD" extension%                                  0447
      jfn _ jrnamf(jfn,$filenm,$"NLD;");                            0448
   %open, etc%                                                      0449
      jworkstid.stfile_open(jfn,$filenm);                           0450
      jworkstid.stpsid_origin;                                      0451
      makeptr(jworkstid, $jwp1);                                    0452
   % set number, mesflg,  %                                        0453
      rfcflg_FALSE;                                                 0454
      IF NOT (FIND jwp1 (C"(J" / "(j"] ^z2 ^z1 ) _2z1 4$5 D ^z3)
      THEN BEGIN                                                    0455
         %bad file, rename with "BAD as ext.%                       0456
         (olbadf): close (jworkstid.stfile:=0);                     0457
         reljfn(jrnamf(0,$filenm,$"BAD;"));                         0458
         REPEAT LOOP;                                               0459
         END;                                                       0460
      *number*_z2 z3;                                               0461
      IF flgs .A 100B THEN deferd _ (*number* = "1234");            0462
      IF NOT deferd AND *number* = "1234" THEN GOTO olbadf;         0463
      secdflag _ FALSE;                                             0464
      % set message flag %                                          0465
         FIND z1 < CH;                                              0466
         CASE READC OF                                              0467
            ='H: mesflg_hcopyv;                                     0468
            ='F, ='M: % let size determine type, unless private %
                                                                    0469
               BEGIN                                                0470
               mesflg _ jtypchk(1000, jworkstid);                   0471
               IF mesflg AND rdprvsts(jwp1.stfile) = $psprivate
               THEN mesflg _ FALSE;                                 0472
               END;                                                 0473
            ='U: secdflag _ 2;                                      0474
            ='S: secdflag _ 1;                                      0475
            ENDCASE %error%;                                        0476
      % set recording flag %                                        0477
         FIND z1;                                                   0478
         CASE READC OF                                              0479
            ='U: recorded _ FALSE;                                  0480
            ENDCASE                                                 0481
               recorded _ TRUE;                                     0482
      %make sure there are 4 semicolons before z1%                  0483
         IF (FIND z1 < C';] ^z4) AND (NOT FIND ";;;") THEN ST z4
         z4 _ ";;;";                                                0484
      %open tjcat if it hasn't been already%                        0485
         IF recorded AND NOT jcstid.stfile THEN jcstid _
         openlock(0, jflname($"tjcat"));                            0486
      IF secdflag = 2 THEN                                          0487
         BEGIN                                                      0488
         %jcat entry not here yet, try to get it%                   0489
         %not implemented yet%                                      0490
         END;                                                       0491
      IF secdflag                                                   0492
         THEN nethcflag _ getnhcflag(inftyp # 0)                    0493
         ELSE                                                       0494
            BEGIN                                                   0495
            FIND jwp1 > (C"Clerk: "] CL/D]) ^z4 _z4 $(L/D) ^z5;
```

```
                                                                    0496
            *clrkstr* _ z4 z5;                                      0497
            %get delivery flag for clerk on this machine%           0498
               %don't use this for now                              0499
                  ckident($clrkstr, $xlit, idfilno);                0500
                  getinlhost($xlit, 0, $z4, $z5);                   0501
                  nethcflag _ CASE lhostn OF                        0502
                     = archost:                                     0503
                        IF (FIND BETWEEN z4 z5 ([*arcstr*])) OR
                        NOT (FIND BETWEEN z4 z5 ([*utilstr*]))
                        THEN TRUE ELSE FALSE;                       0504
                     = utilhost:                                    0505
                        IF FIND BETWEEN z4 z5 ([*utilstr*]) THEN
                        TRUE ELSE FALSE;                            0506
                     ENDCASE 0;                                     0507
               %                                                    0508
               nethcflag _ getnhcflag(inftyp # 0);                  0509
            END;                                                    0510
      IF deferd THEN BEGIN                                          0511
         IF *number* # "1234" THEN GOTO olbadf;                     0512
         getcnum($clrkstr, $number, $" JOURNAL ", 2, 0);            0513
         ST z2 z3 _ *number*;                                       0514
         FIND jwp1 > ([".HJOURNAL="] ["1234"]) ^z3 < 4CH ^z2;       0515
         ST z2 z3 _ *number*;                                       0516
         END;                                                       0517
      IF outremsiteflag AND inftyp # 0 THEN                         0518
         BEGIN                                                      0519
         *remfname* _ "<OUTJOURNAL>R", *number*, ".NLD;";           0520
         IF NOT updtfl(jworkstid.stfile, 1, $remfname ) THEN
         snkerr ($"Journal Remote Update Error");                   0521
         END                                                        0522
      ELSE remfname.L _ 0;                                          0523
      IF NOT secdflag THEN                                          0524
         BEGIN                                                      0525
         setjheader($number, $modstr, idfilno); %fill out journal
         header statement in tejournal or outjournal%               0526
         FIND jwp1 > (["(J" / "(j"] ^z1 ^z3 ) _2z1 _4z3  ['';] ^z2;
                                                                    0527
         *headstr* _ z1 z2;                                         0528
         ST jworkstid _ SF(jworkstid) z3, z2 SE(jworkstid);         0529
         END;                                                       0530
   IF secdflag THEN dsecdoc($number, nethcflag, inftyp#0, jcstid,
   idfilno)                                                         0531
   ELSE CASE mesflg OF                                              0532
      = 1: BEGIN %message%                                          0533
         CASE recorded OF                                           0534
            = TRUE:                                                 0535
               BEGIN                                                0536
               IF NOT (jsavaccess _ access .A accmask[jrnlaccess])
               THEN                                                 0537
                  enablaccess(0, jrnlaccess);                       0538
               IF jnlstid.stfile THEN usednum _ usednum + 20 ELSE
               jnlstid _ findactive(20, jcstid: usednum); %number
               should be number of statements in prospective
               journal document%                                    0539
               entrjdoc(jnlstid, $number,$headstr);                 0540
```

```
            updjcat($number, usednum, $mfname, jcstid); %update
            jcat file%                                          0541
            *tempsr* _ 'J, *number*;                            0542
            jcatentry(IF jdebug THEN $"DUVALL" ELSE $"JOURNAL",
            $mfname,  %file name%  $tempsr, FALSE,
            jcstid,$headstr); %bbranch name%                    0543
            %Now insert Catalog Modification Commands as
            necessary%                                          0544
               IF modstr.L > 0 THEN                             0545
                  BEGIN                                         0546
                  IF (tstid _ namelook(jcstid, $"Catmods")) =
                  endfil THEN                                   0547
                     BEGIN                                      0548
                     IF (tstid _ namelook(jcstid, $"jcatalog"))
                     = endfil THEN badfil(jcstid.stfile);       0549
                     tstid _ cis(tstid, $"(catmods) Catalog
                     Modifiction Commands", $sucdir);           0550
                     END;                                       0551
                  cis(tstid, $modstr, $downstr);                0552
                  END;                                          0553
               END;                                             0554
            ENDCASE % unrecorded message %                      0555
               BEGIN                                            0556
               % assuming work file extension = .NLx %          0557
               FIND SF(*filenm*) ^z1;                           0558
               IF FIND > z1 [".NL"] ^z2 CH ^z3 THEN             0559
                  *mfname* _ z1 z2, 'U, z3 SE(*filenm*)         0560
               ELSE  *mfname* _ *filenm*;                       0561
               END;                                             0562
         distdoc($mfname, IF jdebug THEN $"DUVALL" ELSE
         $"JOURNAL", $number, TRUE, $headstr, idfilno, nethcflag,
         recorded); %distribute it%                             0563
         END;                                                   0564
      ENDCASE BEGIN %non message--mesflg#1%                     0565
         CASE recorded OF                                       0566
            = TRUE:                                             0567
               BEGIN                                            0568
               IF NOT (jsavaccess _ access .A accmask[jrnlaccess])
               THEN                                             0569
                  enablaccess(0, jrnlaccess);                   0570
               %Now open a work file with number as name%       0571
                  IF mesflg # hcopyv THEN                       0572
                     BEGIN                                      0573
                     jdocfile(jcstid, $number, $jworkstid,
                     $dirname); %Update to new file in proper
                     directory%                                 0574
                     END;                                       0575
               %Now update control file%                        0576
                  jcatentry($dirname, $number, $"1", mesflg,
                  jcstid, $headstr);                            0577
                  %Now insert Catalog Modification Commands as
                  necessary%                                    0578
                     IF modstr.L > 0 THEN                       0579
                        BEGIN                                   0580
                        IF (tstid _ namelook(jcstid, $"Catmods"))
                        = endfil THEN                           0581
```

```
                                BEGIN                                   0582
                                IF (tstid _ namelook(jcstid,
                                $"jcatalog")) = endfil THEN
                                badfil(jcstid.stfile);                  0583
                                tstid _ cis(tstid, $"(catmods) Catalog
                                Modifiction Commands", $sucdir);        0584
                                END;                                    0585
                             cis(tstid, $modstr, $downstr);             0586
                             END;                                       0587
                          *dfname* _ *number*;                          0588
                      END;                                              0589
                  ENDCASE % unrecorded file %                           0590
                     BEGIN                                              0591
                     % NOT IMPLEMENTED - NEED STORAGE LOCATION, agreements
                     with archive system %                              0592
                     IF jdebug THEN err($" xnls journal system error");
                                                                        0593
                     END;                                               0594
                  distdoc($dfname, $dirname, $number, mesflg, $headstr,
                  idfilno, nethcflag, recorded); %distribute it%        0595
                  END;                                                  0596
            delpc(jworkstid.stfile,$tempsr);                            0597
            close(jworkstid.stfile := 0);                               0598
            %if we've been working in <outjournal> file then rename it from
            "NLD to "NLR"%                                              0599
               IF remfname.L THEN reljfn(jrnamf(0, $remfname, $"NLR;"));
                                                                        0600
            %rename with ext reflecting the date and release jfn%       0601
               reljfn(jrnamf(0,$filenm, IF recorded THEN $dayext ELSE
               $"NLU;"));                                               0602
            END;                                                        0603
         %Close distfile, tjcat, message file%                          0604
            close (diststid.stfile := 0);                               0605
            close (jcstid.stfile := 0);                                 0606
            closeu(jnlstid.stfile:=0);                                  0607
         RETURN END.
                                                                        0608
% Process distribution file-- online distribtion %                      0609
   (oldist)PROC(flgs,idstr);                                            0610
      %Does the on-line distribution of JOurnal Documents%              0611
      %Bit 40B of flgs to update users' initial files at delivery%      0612
      %Bit 100B: go to operator for input files to process%             0613
      %Bit 200B: only deliver to idents listed in idstr (may contain
      group idents)%                                                    0614
      %Bit 400B: ask operator for document number to distribute%        0615
      LOCAL stid, unrecstid, stid1, tstid, idstid, pcap, adrstid, host,
      ncdstid, jautstid, jdelstid, jinfostid, xstr, wrkstr;             0616
      LOCAL flguif, wierdlock, dnetflag, dolflag, dlinkflag, mailoflag,
      recorded, message, actflag, adstr[40], mhstid;                    0617
      LOCAL TEXT POINTER z1, z2, z3, z4, z5, z6, z7, idstptr;           0618
      LOCAL STRING user[25], notify[55], fnsr[45], tinit[15],
      tempsr[50], filenm[50], stat[30], dnetma[50], spcvw[15],
      jnumstr[10], jrnlfnm[50], hoststr[30], atstr[5];                  0619
      REF idstr, xstr, wrkstr;                                          0620
      %Set up subsy name%                                               0621
         r1 _ oljsbn _ getsbn($"OLJDEL");                               0622
```

```
   !JSYS setnm;                                                    0623
%update users' inital files?%                                      0624
   flguif_ flgs .A 40B;                                            0625
%Enable access to Journal Files%                                   0626
   enablaccess(0, jrnlaccess);                                     0627
%Set up signal and initialise parms%                               0628
   diststid _ hcdstid _ ctlstid _ &xstr _ &wrkstr _ mailoflag _ 0;
                                                                   0629
   ON SIGNAL ELSE                                                  0630
      BEGIN                                                        0631
      sigclose(hcdstid.stfile := 0);                               0632
      sigclose(idstid.stfile := 0);                                0633
      close(jrnlstid.stfile := 0);                                 0634
      closeu(ctlstid.stfile := 0);                                 0635
      IF &xstr # 0 THEN freestring(&xstr:=0,$dspblk);              0636
      IF &wrkstr # 0 THEN freestring(&wrkstr:=0,$dspblk);          0637
      END;                                                         0638
%open id file%                                                     0639
   idstid _ orgstid;                                               0640
   idstid.stfile _ open(0, jflname($"identfile"));                 0641
%process tejournal%                                                0642
   jinsubs(flgs, idstid.stfile);                                   0643
%Update distribution file%                                         0644
   IF updhcdist() = 0 THEN                                         0645
      BEGIN                                                        0646
      %Distribution file empty%                                    0647
      r1 _ nlssbn;                                                 0648
      !JSYS setnm;                                                 0649
      RETURN;                                                      0650
      END;                                                         0651
%If doing restricted distribution, expand the distribution list
idents%                                                            0652
   IF flgs .A 200B THEN iexpdist(&idstr, idstid.stfile);           0653
   FIND SF(*idstr*) ^idstptr;                                      0654
%If doing one document then get number from operator%              0655
   (oldsdn): IF flgs .A 400B THEN BEGIN                            0656
      typeas($"Single document number: ");                         0657
      %get number%                                                 0658
      CASE lookc() OF                                              0659
         IN ["1, "9]: BEGIN                                        0660
            %tnumber($jnumstr);%                                   0661
            *jnumstr* _ "J, *jnumstr*;                             0662
            END;                                                   0663
         ="X, ="x: BEGIN                                           0664
            inpcuc(); %pass the "x"%                               0665
            typeas($" Do all documents now?");                     0666
            IF inpcuc() # CA THEN GOTO oldqui;                     0667
            flgs _ flgs .X 400B;                                   0668
            END;                                                   0669
         ="Q: BEGIN                                                0670
            typeas($"Quit");                                       0671
            GOTO oldqui;                                           0672
            END;                                                   0673
         ENDCASE GOTO oldsdn;                                      0674
      END;                                                         0675
%Now open distribution file%                                       0676
```

```
   hcdstid _ getsub(openlock(0, jflname($"hcdistfile")));          0677
%Initialise the done string (xstr)%                                0678
   &xstr _ getstring(2000, $dspblk);                               0679
% Allocate work string. %                                         0680
   &wrkstr _ getstring(2000, $dspblk);                             0681
*jrnlfnm* _ NULL; %name of currently open message file%            0682
unrecstid _ 0; % temp file in <tejournal>, location of unrecorded
message %                                                          0683
%Now look for people to send stuff to%                            0684
   LOOP % over documents in hcdistfile %                           0685
      BEGIN                                                        0686
      IF hcdstid.stpsid = origin OR discfull(1500) THEN            0687
         % hcdistfile is empty or there are no more documents to
         process %                                                 0688
         EXIT;                                                     0689
      ON SIGNAL                                                    0690
         =sigdsf: EXIT LOOP;                                       0691
         ELSE                                                      0692
            BEGIN                                                  0693
            IF NOT flgs .A 200B THEN hcdstid _ getsuc(hcdstid);
                                                                   0694
            IF flguif OR wierdlock THEN                            0695
               BEGIN                                               0696
               pcap _ enablw();                                    0697
               closeu(ctlstid.stfile := 0);                        0698
               IF pcap # -1 THEN disablw(pcap);                    0699
               END                                                 0700
            ELSE close(ctlstid.stfile := 0);                       0701
            IF mailoflag THEN closemail();                         0702
            REPEAT LOOP;                                           0703
            END;                                                   0704
      IF flgs .A 400B THEN                                         0705
         BEGIN                                                     0706
         % deliver only to documents specified by operator; check
         if this is the one we should be working on %              0707
         CCPOS SF(hcdstid);                                        0708
         *tempsr* _ NULL;                                          0709
         xtrnam($tempsr, $swork, -1);                              0710
         IF *tempsr* # *jnumstr* THEN                              0711
            BEGIN                                                  0712
            hcdstid _ getsuc(hcdstid);                             0713
            REPEAT LOOP;                                           0714
            END;                                                   0715
         typeas($" Found");                                        0716
         END;                                                      0717
      adrstid _ getsub(hcdstid); % Get the first addressee branch
      of this document. %                                          0718
      LOOP % Over all addressees in this currently prime document.
      %                                                            0719
         BEGIN                                                     0720
         IF flgs .A 200B THEN                                      0721
            BEGIN %deliver only to idents in idstptr%              0722
            IF NOT FIND idstptr $NP ^z1 1$LD ^idstptr THEN EXIT;
                                                                   0723
               % no more idents had been specified by the
               operator%                                           0724
```

```
      *fnsr* _ z1 idstptr;                                          0725
      adrstid _ hcdstid;                                            0726
      lookup($adrstid, $fnsr, seqname);                             0727
         % Look for things to be deliverd to this user; if
         none, go on to the next. %                                 0728
      IF adrstid = endfil THEN REPEAT LOOP;                         0729
      END                                                           0730
   ELSE                                                             0731
      BEGIN                                                         0732
      IF adrstid = hcdstid THEN EXIT;                               0733
         % There are no more addressees for this document;
         go on to the next document. %                              0734
      CCPOS SF(adrstid);                                            0735
      *fnsr* _ NULL;                                                0736
      xtrnam($fnsr, $swork, -1);                                    0737
      END;                                                          0738
   % At this point, FNSR has the ident of the user whose
   mail is beig processed. %                                        0739
   IF flgs .A 400B AND getup(adrstid) # hcdstid THEN                0740
      EXIT LOOP; %only deliver hcdstid branch%                      0741
   IF discfull(1400) THEN EXIT 2;  %mainly to check load
   average%                                                         0742
   IF fnsr.L # 0                                                    0743
      AND  ((dolflag _ donline(adrstid :actflag)) .V
      (dnetflag _ dnetdel(adrstid)) # 0)                            0744
      AND (NOT dolflag OR dolflag _ (NOT FIND SF(*xstr*) [SP
      *fnsr* SP]) OR dnetflag)                                      0745
      AND  ckident($fnsr, $xlit, idstid.stfile)                     0746
      THEN                                                          0747
         BEGIN  %Try to distribute it%                              0748
         *hoststr* _ NULL;                                          0749
         IF dolflag THEN *xstr* _ *xstr*, *fnsr*, SP; %mark
         him as being tried%                                        0750
         *tinit* _ *fnsr*; %save off initials%                      0751
         %Now see if we can open his control (initial) file%
                                                                    0752
            luser($xlit, $tempsr, 0, 0);                            0753
            *user* _ *tempsr*; %save off his username % 0754
            IF dolflag AND NOT (ctlstid _ opnctfile($fnsr,
            $tempsr : wierdlock%locked by user flag%)) THEN
                                                                    0755
               dolflag _ FALSE;                                     0756
            IF NOT dolflag AND NOT dnetflag THEN                    0757
               GOTO nodo; % no online delivery necessary. %
                                                                    0758
            %By here, ctl file and/or sequential file is
            open%                                                   0759
            jautstid _ jdelstid _ jinfostid _ 0;                    0760
         %Now start to give him the documents%                      0761
            stid _ adrstid; % so we can get the next user to
            be delivered to %                                       0762
            LOOP % Over all documents to this user. %               0763
               BEGIN                                                0764
               %Get stid of branch in initial file to put it
               in%                                                  0765
                  IF dolflag THEN                                   0766
```

```
                BEGIN                                   0767
                IF FIND SF(stid) [*Author Copy*] THEN
                                                        0768
                   BEGIN                                0769
                   IF jautstid = 0 THEN                 0770
                      IF (jautstid _ namelook(ctlstid,
                      $"author")) = endfil THEN         0771
                         csetnsta((jautstid _
                         cis(ctlstid, $"Author
                         (authored journal documents)",
                         $sucdir)), 0, 0 );             0772
                   ctlstid _ jautstid;                  0773
                   END                                  0774
                ELSE                                    0775
                   BEGIN                                0776
                   dolflag _ dolflag .V 2; %flag for
                   stuff to "Journal" branch%           0777
                   IF jdelstid = 0 THEN                 0778
                      IF (jdelstid _ namelook(ctlstid,
                      $"Journal")) = endfil THEN        0779
                         csetnsta((jdelstid _
                         cis(ctlstid, $"Journal
                         documents (most recent
                         first)", $sucdir)), 0, 0 );
                                                        0780
                   ctlstid _ jdelstid;                  0781
                   END;                                 0782
                END;                                    0783
             stid1 _ getup(stid); % head of branch%     0784
             %Set up link/location pointers%            0785
                dlinkflag _ FALSE;                      0786
                recorded _ TRUE;                        0787
                message _ FALSE;                        0788
                CCPOS SF(stid1);                        0789
                IF FIND [*Hard Copy--Location: *] ^z1 [*;]
                ^z2 THEN                                0790
                   GOTO uselnk                          0791
                ELSE FIND [*)] [*(] ^z1 _z1 [*)] ^z2;
                %Pointers to link%                      0792
                dlinkflag _ TRUE;                       0793
                lnkpspc(1, $z1, $tempsr, $filenm, $stat,
                $spcvw, $adstr);                        0794
                CASE *stat*[1] OF                       0795
                   = *J:                                0796
                      *filenm* _ *filenm*, ".NLS";      0797
                   ENDCASE recorded _ FALSE;            0798
                IF FIND BETWEEN z1 z2                   0799
                   ([*JRNL*/*DUVALL*]%experimental system
                   only%) THEN                          0800
                   BEGIN                                0801
                   message _ TRUE;                      0802
                   IF recorded THEN                     0803
                      BEGIN                             0804
                      IF *filenm* # *jrnlfnm* THEN      0805
                         BEGIN                          0806
                         close(jrnlstid.stfile := 0);   0807
```

```
                    jrnlstid _ openlock(0, $filenm);
                                                        0808
                    *jrnlfnm* _ *filenm*;               0809
                    END;                                0810
                 END                                    0811
              ELSE % unrecorded %                       0812
                 BEGIN                                  0813
                 unrecstid _ orgstid;                   0814
                 unrecstid.stfile _ open(0, $filenm);
                                                        0815
                 FIND SF(unrecstid) ^z5;                0816
                 z4 _ getsub(z5); % location of
                 message plex %                         0817
                 GOTO uselnk;                           0818
                 END;                                   0819
              FIND SF(jrnlstid) ^z3 ;                   0820
              specreg($stat, nametyp, $z3);             0821
              %z5 set for network delivery%             0822
                 z5 _ z3;                               0823
                 z5[1] _ z3[1];                         0824
              IF z3 = endfil THEN GOTO uselnk;          0825
              z4 _ getsub(z3); % location of message
              plex %                                    0826
              dlinkflag _ FALSE;                        0827
              IF dolflag AND NOT olhed(stid, message,
              z3, $z1, $z2, recorded, &wrkstr) THEN
                                                        0828
                 GOTO uselnk; %format citation into
                 lit (and wrkstr if there is a
                 comment). go elsewhere if
                 unsuccessful%                          0829
              END                                       0830
           ELSE                                         0831
              BEGIN %Put out the link%                  0832
              % handle unrecorded files here %          0833
              (uselnk):                                 0834
              %format citation into lit (and wrkstr
              if there is a comment%                    0835
                 IF dolflag THEN olhed(stid, message,
                 z1, $z1, $z2, recorded, &wrkstr);
                                                        0836
                 z5 _ z1; %for net deliv.%              0837
                 z5[1] _ z1[1];                         0838
              END;                                      0839
        %put it out%                                    0840
           IF dolflag THEN                              0841
              BEGIN                                     0842
              % citation %                              0843
                 tstid _ cis (ctlstid, $lit,
                 $downstr);                             0844
              IF message AND (z3 # endfil) THEN         0845
                 % message %                            0846
                 BEGIN                                  0847
                 mhstid _ cis(tstid, $"Message:",
                 $downstr);                             0848
                 ccopgro (mhstid, down, z4,
```

```
                getail(z4), 0, $" ");                    0849
                END;                                     0850
             IF wrkstr.L > 0 THEN                        0851
                % comment %                              0852
                cis(tstid, &wrkstr, $downstr);           0853
             IF unrecstid.stfile THEN                    0854
                close(unrecstid.stfile:=0); % temp
                location of unrecorded messages %
                                                         0855
             %fix up hcdistfile %                        0856
                IF FIND SE(stid) < [*] > ["
                do-ola"/"do-ol"] ^z7 < [SP] ^z6 THEN
                                                         0857
                   ST z6 z7 _ " ol-done";                0858
             END;                                        0859
          IF dnetflag AND hoststr.L=0 THEN               0860
             BEGIN                                       0861
             getihost($xlit, $hoststr, 0, 0);            0862
             getinma($xlit, $dnetma, 0, 0);              0863
             END;                                        0864
          IF dnetflag AND (dnetflag _ mailoflag _
          openmail($dnetma, $hoststr,
          $"OUTJOURNAL")) THEN                           0865
             BEGIN                                       0866
             nwkhed(stid, message, z5, $z1, $z2,
             recorded, &wrkstr, idstid.stfile);          0867
             mailstring($lit, 1);                        0868
             IF wrkstr.L > 0 THEN                        0869
                mailstring(&wrkstr, 2);                  0870
             IF message AND (z3 # endfil) THEN           0871
                BEGIN                                    0872
                mailstring($"Message:", 2);              0873
                mailgroup(z4, getail(z4), 3);            0874
                END;                                     0875
             closemail();                                0876
             mailoflag _ 0;                              0877
             % set up mailer (?) for network
             distribution. %                             0878
                pokemailer($"OUTJOURNAL");               0879
             %fix up hcdistfile %                        0880
                IF FIND SE(stid) < [*] > ["
                do-net"] ^z2 < [SP] ^z1 THEN             0881
                   ST z1 z2 _ " net-done";               0882
             END;                                        0883
       %Now proceed to next document for this user%
                                                         0884
          IF donline THEN LOOP                           0885
             BEGIN                                       0886
             z1 _ stid;                                  0887
             lookup($z1, $tinit, seqname);               0888
             IF z1 = endfil THEN EXIT 2; %done with
             this user; may have other documents for
             other users to process%                     0889
             stid _ z1;                                  0890
             dnetflag _ dnetdel(stid);                   0891
             IF donline(stid : actflag)  THEN EXIT;
```

```
                        %Do this one%                                   0892
                        END %of search loop%                            0893
                    ELSE EXIT; %not doing online deliv%                 0894
                END; %of main dstribution loop for a user%
                                                                        0895
            %Now close his control file%                                0896
              IF dolflag THEN                                           0897
                BEGIN                                                   0898
                IF (dolflag .A 2) THEN %mark fdb with
                "journal mail" flag%                                    0899
                  BEGIN                                                 0900
                  pcap _ enablw();                                      0901
                  r1 _ 24B6 +
                  [flntadr(ctlstid.stfile)].florig;                     0902
                  r3 _ r2 _ 0;                                          0903
                  r2.ojdelf _ r3.ojdelf _ 1;                            0904
                  !JSYS chfdb;                                          0905
                  IF pcap # -1 THEN disablw(pcap);                      0906
                  END;                                                  0907
                IF flguif %update initials files%  OR
                wierdlock THEN                                          0908
                  BEGIN                                                 0909
                  pcap _ enablw();                                      0910
                  closeu(ctlstid.stfile := 0);                          0911
                  IF pcap # -1 THEN disablw(pcap);                      0912
                  END                                                   0913
                ELSE close(ctlstid.stfile := 0);                        0914
                END;                                                    0915
            END; %of this user%                                         0916
          (nodo): IF NOT flgs .A 200B THEN                              0917
            % Get the next addressee for the currently prime
            document and try to process all of his mail. %              0918
            adrstid _ getsuc(adrstid);                                  0919
          END; %Of this document%                                       0920
        IF NOT flgs .A 600B THEN                                        0921
          % get the next document to be processed %                     0922
          hcdstid _ getsuc(hcdstid);                                    0923
        END; % of loop over documents in hcdistfile. %                  0924
      IF flgs .A 400B THEN                                              0925
        GOTO oldsdn; % Documents to be delivered were specified by
        operator; go back to see if there were more specified. %
                                                                        0926
    %Now wrap it up%                                                    0927
      (oldgui):                                                         0928
      close(hcdstid.stfile := 0);                                       0929
      close(idstid.stfile := 0);                                        0930
      close(jrnlstid.stfile := 0);                                      0931
      *jrnlfnm* _ NULL;                                                 0932
      closeu(ctlstid.stfile := 0);                                      0933
      freestring (&xstr, $dspblk);                                      0934
      freestring (&wrkstr, $dspblk);                                    0935
    RETURN                                                              0936
    END.
                                                                        0937
% Process distribution file-- offline hardcopy distribtion %            0938
  (autoncd)PROC(rundet);                                                0939
```

```
%Run hard copy automatically.  If rundet is TRUE, set up primary
output file and run detached.%                                          0940
LOCAL det, pjfn;                                                        0941
LOCAL TEXT POINTER z1, z2;                                              0942
LOCAL STRING tempsr[50];                                                0943
%Initialise parms and set up SIGNAL%                                    0944
   pjfn _ det _ 0;                                                      0945
   ON SIGNAL ELSE                                                       0946
      BEGIN                                                             0947
      %check for disc full termination%                                 0948
         CASE sysgnl OF                                                 0949
            = sigdsf: BEGIN                                             0950
               dismes (1, sysmsg);                                      0951
               jdquit();                                                0952
               END;                                                     0953
            ENDCASE;                                                    0954
      %Now type message and restore prinary ouput file to normal%
                                                                        0955
         IF rundet  AND pjfn THEN                                       0956
            BEGIN                                                       0957
            %Fix up primary output %                                    0958
               r1 _ 4B5;                                                0959
               !JSYS gpjfn;                                             0960
               r2.RH _ r2.LH;                                           0961
               !JSYS spjfn;                                             0962
            sysclose(pjfn := 0, 0);                                     0963
            END;                                                        0964
         %type a message indicating automatic stop%                     0965
            *tempsr* _ NULL;                                            0966
            !JSYS gtad;                                                 0967
            dtfrmt(r1, $tempsr);                                        0968
            *tempsr* _ *tempsr*, " Auto Hard Copy stop";                0969
            IF det  = -1 THEN %Running detatched...type out
            message to logging tty%                                     0970
               specttyout(0, $tempsr);                                  0971
      RETURN;                                                           0972
      END;                                                              0973
%First type a message indicating automatic startup%                     0974
   *tempsr* _ NULL;                                                     0975
   !JSYS gtad;                                                          0976
   dtfrmt(r1, $tempsr);                                                 0977
   *tempsr* _ *tempsr*, " Auto Hard Copy Start";                        0978
   !JSYS gjinf;                                                         0979
   IF (det _ r4) = -1 THEN %Running detatched...type out message
   to logging tty%                                                      0980
      specttyout(0, $tempsr)                                            0981
   ELSE typeas($tempsr);                                                0982
%Operator is "XXX"%                                                     0983
   *opstr* _ "XXX";                                                     0984
%Close initial file%                                                    0985
   close([tda].dacsp.stfile:= 0);                                       0986
%Open file for primary output%                                          0987
   IF rundet THEN                                                       0988
      BEGIN                                                             0989
      *tempsr* _ NULL;                                                  0990
      !JSYS gtad;                                                       0991
```

```
            dtfrmt(r1, $tempsr);                                        0992
            IF FIND SF(*tempsr*) [SP] ^z1 _z1 SE(z1) ^z2  THEN          0993
               ST z1 z2 _ NULL;                                         0994
            *tempsr* _ *tempsr*, "autohcd.out";                         0995
            pjfn _ setupop(0, $tempsr, 4, IF det = -1 THEN 0 ELSE -1);
                                                                        0996
            END;                                                        0997
         %Now set up and run hard copy%                                 0998
            inithcd();                                                  0999
            printall(1, mastacprintflag, 0); %Distribution and Collections%
                                                                        01000
         %Now quit%                                                     01001
            jdquit();                                                   01002
         callsig(0, 0);%cause signal to reset prnary output and return%
                                                                        01003
         END.
                                                                        01004
% Entry support-- expand Journal header%                                01005
   (setjheader)PROC(number, modstr, idflnm);                            01006
      %*setjheader*                                                     01007
         This procedure  basically converts an abbreviated Journal
         header [which is the result of Submit] into a full-fledged
         final format Journnal header.                                  01008
         Along the way, it:                                             01009
            Fills out the Author and Distribution fields.  This entails
            expanding all groups as necessary, and from the resulting
            list  of expanded group and individual idents, insert the
            names of all individuals/groups into the proper place in the
            header (Specifically before the '/ preceding the identlist)
                                                                        01010
            It fills out the subcollection field with the actual
            contents of the field, any default subcollections specified
            by submission parameters (e.g. NWG/RFC, NIC), and the names
            of any groups in the distributiion list.  Redundant entries
            are deleted.                                                01011
            If there are any Obsoletes or Updates in the header, it
            makes up catalog modification requests for the obsoleted and
            updated documents, and returns these in the strig modstr
            (the calling procedure will put them unnder a branch in the
            Journal catalog, and sometime in the future the wholle mess
            willl b submitted to the catalog system for catalog citation
            maintenance.                                                01012
         %                                                              01013
      LOCAL TEXT POINTER idptr, z1, z2;                                 01014
      LOCAL idfnum, stid;                                               01015
      LOCAL STRING adsubcols[100], oldsubcols[300], tempsr[30];         01016
      REF number, modstr;                                               01017
      %Number is Journal Number, modstr will be used for returning any
      catalog manipulation commands for other entries%                  01018
      %This routine fills out the journal header%                       01019
      IF idflnm = 0 THEN                                                01020
         BEGIN                                                          01021
         idfnum _ 0;                                                    01022
         ON SIGNAL ELSE sigclose(idfnum := 0);                          01023
         idfnum _ open(0, jflname($"identfile"));                       01024
         END                                                            01025
```

```
   ELSE idfnum _ idflnm;                                                01026
   %Now fill out name fields%                                           01027
      %author field%                                                    01028
         *lit* _ NULL;                                                  01029
         IF NOT (FIND jwp1 > [*"Author(s): "] [*/] ^idptr) THEN         01030
            snkerr($"Bad header statement");                            01031
         intids(0);                                                     01032
         LOOP                                                           01033
            BEGIN                                                       01034
            IF NOT getids($idptr, $lit, 1, idfnum) THEN EXIT; %reads
            idents from jwp1 (up to ";) and puts info into $lit
            (appended)...returns FALSE if no idents left%               01035
            *lit* _*lit*,", ";                                          01036
            END;                                                        01037
         IF lit.L > 0 THEN lit.L _ lit.L-2; %ignore last comma%         01038
         CCPOS SF(jwp1);                                                01039
         FIND [*"Author(s): "] ^jwp1;                                   01040
         ST jwp1 _ SF(jwp1) jwp1, *lit*, jwp1 SE(jwp1);                 01041
   IF NOT idflnm THEN close(idfnum := 0);                               01042
   %Now fix up Sub-collection Field to reflect RFC, Distribution
   Groups%                                                              01043
      IF FIND SF(jwp1) ([*"RFC# "] 3$D) THEN rfcflg _ 2;                01044
      IF FIND SF(jwp1) ^jwp1 SF(*[galjsubcol($xlit,idfnum)]*) ^z1
      THEN                                                              01045
         BEGIN                                                          01046
         *oldsubcols* _ z1 SE(z1);                                      01047
         setjsubcol($oldsubcols);                                       01048
         END;                                                           01049
   %Now build any catalog modification commands dictated by parameter
   fields in the header%                                                01050
      IF &modstr THEN                                                   01051
         BEGIN                                                          01052
         %Obsoletes%                                                    01053
            getjobsoletes($xlit); % Nulls out xlit if none. %           01054
            IF xlit.L THEN                                              01055
               *modstr* _ "Obsoletes Document(s): ", *xlit*, EOL
                                                                        01056
            ELSE *modstr* _ NULL;                                       01057
         %Updates%                                                      01058
            getjupdates($xlit); % Nulls out xlit if none. %             01059
            IF xlit.L THEN                                              01060
               *modstr* _ *modstr*, "Updates Document(s): ", *xlit*;
                                                                        01061
         %Check to see if entries are not NULL%                         01062
            IF modstr.L THEN                                            01063
               *modstr* _ *number*, SP, *modstr*;                       01064
         END;                                                           01065
   RETURN END.
                                                                        01066
% Build J-catalog entries %                                            01067
   (updjcat)PROC(number, numused, linkstr, jcstid);                     01068
      %This routine updates the jcat file, where numused is the total
      number of statements used in the active file.  It is expected that
      number contains tte journal number of the document just entered%
                                                                        01069
      %jcstid is assumed to be the stid of the active file statement%
```

```
                                                                    01070
   LOCAL STRING tempsr[10];                                         01071
   LOCAL TEXT POINTER z1, z2;                                       01072
   REF number, linkstr;                                             01073
   %ton 7%                                                          01074
   IF (jcstid _ namelook(jcstid, $"ACTIVE")) = endfil THEN
   snkerr($"Bad Jcat File");                                        01075
   IF NOT FIND SF(jcstid) ["Used = "] $NP ^z1 1$D ^z2 THEN          01076
      snkerr($"Bad Jcat File");                                     01077
   *tempsr* _ STRING(numused); %Get around L10 Compiler glitch%      01078
   ST z1 _ SF(z1) z1, *tempsr*, z2 SE(z2), " ", *number*;           01079
   %Now save off name for distribution%                             01080
      IF NOT FIND SF(jcstid) ["] ("] ^z1 ['")] ^z2 _z2 THEN         01081
         snkerr($"Bad Jcat File");                                  01082
      *linkstr* _ z1 z2;                                            01083
   %toff 7%                                                         01084
   RETURN;                                                          01085
   END.

                                                                    01086
(jcatentry)PROC(astr1, astr2, astr3, hcflag, jcstid, headstr);      01087
   LOCAL stid, xstid;                                               01088
   %make an entry in the catalog part of the jcat file.             01089
      astr1 is user name, astr2 is file name for file, and astr3 is
      branch name%                                                  01090
   LOCAL TEXT POINTER z1, z2;                                       01091
   REF astr1, astr2, astr3, headstr;                                01092
   %ton 8%                                                          01093
   stid _ jcstid;                                                   01094
   IF (xstid _ namelook(stid, $"JCATALOG"))= endfil THEN snkerr($"Bad
   Jcat File");                                                     01095
   xstid _ ccopsta(xstid, levdown, jworkstid, FALSE, 0);            01096
   FIND SF(xstid) ^z1 [">"] ["","] ['";] ^z2;                       01097
   ST z1 z2 _ *headstr*;                                            01098
   stid _ xstid;                                                    01099
   xstid _ cis(xstid, &astr1, $downstr);                            01100
   IF hcflag THEN                                                   01101
      BEGIN                                                         01102
      CCPOS SF(stid);                                               01103
      FIND ["Location: "] ^ z1 ['";] ^z2;                           01104
      ST xstid _ "Location of Document: ",  z1 z2;                  01105
      %toff 8%                                                      01106
      END                                                           01107
   ELSE                                                             01108
      ST xstid _ "Link to document: (", *astr1*, ', , *astr2*, ', ,
      *astr3*, ":w) ";                                              01109
   %toff 8%                                                         01110
   RETURN END.

                                                                    01111
(jdoctile)PROC(jcstid, fname, jwstid,  dirnam);                     01112
   %Given the stid of jcat, looks for a directory name statement, and
   checks that there is room for the indicated file in the directory.
                                                                    01113
      If everything is ok, it updates the file passed as jwstid into
      a new file in the new directory named fname.                  01114
      If directory is full, it looks for an ndirectory statement, and
      activates it, returning as above.                             01115
```

```
      It returns the name of the directory which was used          01116
      It signals if something goes wrong%                          01117
   LOCAL jdocstid;                                                 01118
   LOCAL TEXT POINTER z1, z2, z3, z4;                              01119
   LOCAL STRING tfname[50];                                        01120
   REF fname, dirnam, jwstid;                                      01121
   %First look for directory statment in JCAT%                     01122
      IF (jcstid _ namelook(jcstid, $"DIRECTORY")) = endfil THEN
      snkerr($"Directory list exhausted");                         01123
      IF NOT FIND SF(jcstid) ["Name: "] $NP ^z1 [";] < CH ^z2 > THEN
      snkerr($"Bad Jcat File");                                    01124
      *dirnam* _ z1 z2;                                            01125
   %Now open file%                                                 01126
      ON SIGNAL ELSE                                               01127
         BEGIN                                                     01128
         %Delete current directory name, and look for ndirectory
         statements%                                               01129
            FIND SF(jcstid) ['(] $NP ^z1;                          01130
            ST z1 z1 _ 'F;                                         01131
            IF (jcstid _ namelook(jcstid, $"NDIRECTORY")) = endfil
            THEN                                                   01132
               SIGNAL(-1, $"Directory list exhausted");            01133
            FIND SF(jcstid) ['(] $NP ^z1 CH ^z2;                   01134
            ST z1 z2 _ NULL;                                       01135
         RETURN(jdocfile(jcstid, &fname, &jwstid, &dirnam));       01136
         END;                                                      01137
      *tfname* _ '<, *dirnam*, '>, *fname*, ".NLS", EOL;           01138
      IF NOT updtfl(jwstid.stfile, 1, $tfname) THEN snkerr($"Fatal
      Journal File Error");                                        01139
      setaccess(jwstid.stfile, jrnlaccess); %no toucheeee%         01140
      RETURN(jdocstid);                                            01141
   END.
                                                                   01142
% Hard copy delivery utility procedures %                         01143
  (inithcd)PROC; % Initialize for hard copy delivery. %            01144
      LOCAL TEXT POINTER z1, z2;                                   01145
      %Initialise Parameters for Hard Copy Distribution%           01146
         %Global stids and  file variables%                        01147
            docjfn _ hdrjfn _ docstrt _ lastfnum _ lptused _ lptjfn _
            idfile _ prntfg _ 0;                                   01148
         %set up subsystem name for statistics%                    01149
            r1 _ oljsbn _ getsbn($"HCJDEL");                       01150
            !JSYS setnm;                                           01151
      %Now see which output processor and printer file we are to use%
                                                                   01152
         IF jdebug THEN                                            01153
            BEGIN                                                  01154
            %Debugging--use normal output processor and print file xlpt%
                                                                   01155
            *ptstr* _ "XLPT.TXT";                                  01156
            *oprocn* _ *opname*;                                   01157
            END                                                    01158
         ELSE                                                      01159
            BEGIN % real system..normal output processor and lpt%  01160
            *ptstr* _ NULL;                                        01161
            !JSYS gtad;                                            01162
```

```
        dtfrmt(r1, $ptstr);                                          01163
        IF FIND SF(*ptstr*) [SP] ^z1 _z1 SE(z1) ^z2 THEN ST z1 z2 _
        NULL;                                                        01164
        *ptstr* _ "<ARCDOCUMENTATION>", *ptstr*, "LPT.TXT";          01165
        *oprocn* _ *opname*;                                         01166
        END;                                                         01167
   %Now update hcdistfile from distfile%                             01168
      IF updhcdist() = 0 THEN                                        01169
        BEGIN                                                        01170
        typeas( $"Distribution File Empty");                         01171
        jdquit();                                                    01172
        SIGNAL( 0, $"Distribution File Empty");                      01173
        END;                                                         01174
   RETURN;                                                           01175
   END.
                                                                     01176
(printall)PROC(distribution, collections, nic);                      01177
   %This procedure prints all of the indicated copies of all
   documents in the hcdistfile%                                      01178
   LOCAL hcdstid, stid;                                              01179
   hcdstid _ 0;                                                      01180
   ON SIGNAL ELSE                                                    01181
      BEGIN                                                          01182
      sigclose(hcdstid.stfile := 0);                                 01183
      lptclose();                                                    01184
      END;                                                           01185
   hcdstid _ openlock(0, jflname($"hcdistfile"));                    01186
   pfilnum _ 1;                                                      01187
   stid _ getsub(hcdstid);                                           01188
   WHILE stid # hcdstid DO                                           01189
      BEGIN                                                          01190
      printdoc(stid:= getsuc(stid), distribution, collections, nic);
                                                                     01191
      BUMP pfilnum;                                                  01192
      END;                                                           01193
   close(hcdstid.stfile :=0);                                        01194
   lptclose();                                                       01195
   RETURN;                                                           01196
   END.
                                                                     01197
(printdoc)PROC(hcdstid, distribution, collections, nic);             01198
   %Print out the various flavors of the document identified by
   distfile entry hcdstid.                                           01199
      Flags indicate which copies to print%                          01200
   LOCAL dwstid, stid, wrkstr;                                       01201
   LOCAL TEXT POINTER z1, z2;                                        01202
   LOCAL STRING jnum[5];                                             01203
   REF wrkstr;                                                       01204
   dwstid _ 0;                                                       01205
   prntfg _ TRUE;                                                    01206
   % allocate work string. %                                        01207
      &wrkstr _ getstring(2000, $dspblk);                            01208
   ON SIGNAL ELSE                                                    01209
      BEGIN                                                          01210
      close(dwstid.stfile := 0);                                     01211
      IF &wrkstr THEN freestring( &wrkstr:=0, $dspblk);              01212
```

```
      CASE sysgnl OF                                                    01213
         = -4: NULL;                                                    01214
         ENDCASE                                                        01215
            BEGIN %Not called to rubout%                                01216
            dismes(1, sysmsg);                                          01217
            IF sysgnl # sigdsf THEN RETURN;                             01218
            END;                                                        01219
      END;                                                              01220
   %get document number into jnum, and type it to the operator%         01221
      FIND SF(hcdstid) ^z1;                                             01222
      getjnm($z1, $jnum);                                               01223
      crlf();                                                           01224
      typeas($"Document #");                                            01225
      typeas($jnum);                                                    01226
      crlf();                                                           01227
   %set up document for printing%                                       01228
      %Get author names and addresses for address page%                 01229
         getdauth(hcdstid, &wrkstr);                                    01230
      %Set up te dwork file%                                            01231
         dwstid _ getdwork(hcdstid, &wrkstr);                           01232
   %Do distribution copies first%                                       01233
      IF distribution THEN                                              01234
         BEGIN                                                          01235
         stid _ getsub(hcdstid);                                        01236
         WHILE stid # hcdstid DO                                        01237
            pdistcopy(dwstid, stid := getsuc(stid), $jnum, &wrkstr);
                                                                        01238
         END;                                                           01239
   % deallocate work string. %                                         01240
      freestring( &wrkstr:=0, $dspblk);                                 01241
   %do system copies of public documemts%                               01242
      IF NOT getcacc (hcdstid, 0, 0, 0) THEN                            01243
         BEGIN                                                          01244
         %Do collection copies%                                         01245
            IF collections AND NOT FIND SF(hcdstid) [EOL "Secondary
            Distribution Copy"] THEN                                    01246
               BEGIN                                                    01247
               collcopy(dwstid, hcdstid, $"Master", $jnum);             01248
               END;                                                     01249
         %Do NIC copies%                                                01250
            IF nic AND FIND SF(hcdstid) ["Sub-Collections: "] ^z1
            [";] ^z2 BETWEEN z1 z2(["NIC"]) THEN                        01251
               BEGIN                                                    01252
               collcopy(dwstid, hcdstid, $"NIC", $jnum);                01253
               nicsitecopies(dwstid, hcdstid);                          01254
               END;                                                     01255
         END;                                                           01256
   close(dwstid.stfile := 0);                                           01257
   RETURN;                                                              01258
   END.
                                                                        01259
(printnum)PROC(numstr, distribution, collections, nic);                 01260
   %Print the document indicated by numstr if there is an entry for
   it in the hcdistfile%                                                01261
   LOCAL hcdstid, stid;                                                 01262
   LOCAL STRING tempsr[10];                                             01263
```

```
   REF numstr;                                                    01264
   hcdstid _ 0;                                                   01265
   ON SIGNAL ELSE sigclose(hcdstid.stfile := 0);                  01266
   *tempsr* _ 'J, *numstr*;                                       01267
   hcdstid _ openlock(0, jflname($"hcdistfile"));                 01268
   IF (stid  _ namelook(hcdstid, $tempsr))= endfil THEN err($"No Such
   document in distfile");                                        01269
   pfilnum _ 1;                                                   01270
   printdoc(stid, distribution, collections, nic);                01271
   lptclose();                                                    01272
   close(hcdstid.stfile := 0);                                    01273
   RETURN;                                                        01274
   END.
                                                                  01275
(pdistcopy)PROC(dwstid, adrstid, jnumber, wrkstr);                01276
   %Print a copy of the documen indicated by dwstid for the peson
   addressed by adrstid if it should be done, and mark entry adrstid
   as having been printed%                                        01277
   LOCAL STRING adrstr[700];                                      01278
   LOCAL TEXT POINTER z1, z2, z3, z4;                             01279
   REF jnumber, wrkstr;                                           01280
   IF discfull(1300) THEN SIGNAL(sigdsf, $"*** TERMINATED: DISC FULL
   ***"); %check disc, also wait if load average high%            01281
   CCPOS SF(adrstid);                                             01282
   xtrnam($adrstr, $swork, '(, ') );                              01283
   ckident($adrstr, &wrkstr, 0);                                  01284
   IF phcopy(adrstid) THEN %A copy should be printed for this user%
                                                                  01285
      BEGIN                                                       01286
      %Get his name and address%                                  01287
         getinam(&wrkstr, 0, $z1, $z2);                           01288
         laddress(&wrkstr, $adrstr, 0, 0, 0);                     01289
         *adrstr* _ z1 z2, EOL, *adrstr*, ".LMS=60;.GCR=5;",
         *jnumber*, ".LMS=0;.GCR;";                               01290
         %Now look for possible comments%                         01291
            IF FIND SF(adrstid) ([Note: "] ^z1 ["];"] ^z2  %New
            format%                                               01292
               /[".GCR=5;.GCR=5;"] ^z1 [EOL EOL] ^z2) _2z2 %Old
               Format%                                            01293
               THEN *adrstr* _ *adrstr*, ".GCR=4;.GCR=5;", z1 z2,
               EOL;                                               01294
      %Now print it%                                              01295
         printit(dwstid, $adrstr, pfilnum);                       01296
      %Now tag statement as having been printed%                  01297
         IF FIND SE(adrstid) < ['*] > [" do-hc"] ^z2 < [SP] ^z1 THEN
         ST z1 z2 _ " hc-done-", *opstr*;                         01298
      END;                                                        01299
   RETURN;                                                        01300
   END.
                                                                  01301
(collcopy)PROC(dwstid, hcdstid, astr, jnum);                      01302
   %Print a 'Collection Copy' For collection identified by astr if it
   has not already been done.  Tag statement hcdstid accordingly%
                                                                  01303
   LOCAL STRING tempsr[100];                                      01304
   LOCAL distexf, distid;                                         01305
```

```
   LOCAL TEXT POINTER p1;                                              01306
   REF jnum, astr;                                                     01307
   IF discfull(1300) THEN SIGNAL(sigdsf, $"*** TERMINATED: DISC FULL
   ***"); %check disc, also wait if load average high%                 01308
   *tempsr* _ *astr*, " Copy Printed";                                 01309
   IF NOT FIND SF(hcdstid) [*tempsr*] THEN %It has not been printed%
                                                                       01310
      BEGIN                                                            01311
      IF FIND SF(*astr*) "Master" SF(dwstid) ["Distribution:"] ['/]
      ^p1 THEN %put out a couple of copies of the expanded
      distribution%                                                    01312
         BEGIN                                                         01313
         *tempsr* _ *jnum*, " Distribution";                           01314
         distid _ cis(dwstid, $tempsr, $downstr);                      01315
         distexf _ TRUE;                                               01316
         expdist(distid, $p1);                                         01317
         poutput (distid, $hdrjfn, $hdrwfn);                           01318
         chklpt();                                                     01319
         copylpt(FALSE, hdrjfn, 0, -1);                                01320
         cdelgro(distid, distid, FALSE, 0); %delete distribution
         branch%                                                       01321
         END;                                                          01322
      *tempsr* _ *astr*, " Copy.LMS=60;.GCR=5;", *jnum*,
      ".LMS=0;.GCR;";                                                  01323
      printit(dwstid, $tempsr, pfilnum);                               01324
      *tempsr* _  *astr*, " Copy";                                     01325
      tagjdel(hcdstid, $tempsr);                                       01326
      END;                                                             01327
   RETURN;                                                             01328
   END.
                                                                       01329
(printit)PROCEDURE(stid, astr, fnum);                                  01330
   LOCAL cntr, newdoc;                                                 01331
   LOCAL TEXT POINTER z1, z2, z3;                                      01332
   REF astr;                                                           01333
   %Replace current name statement if astr # 0%                        01334
      IF &astr # 0 THEN                                                01335
         BEGIN                                                         01336
         CCPOS SF(stid);                                               01337
         IF NOT FIND ["LMS=24;
         "] ^z1 ["
         .PEL;"] [EOL] ^ z3 < [EOL]  [EOL] ^z2 >  THEN                 01338
            SIGNAL(2, $"Illegal Header in Printit");                   01339
         ST stid _ SF(stid) z1,*astr*,z2 z3, ".PSW=0;.HLT;", z3
         SE(stid);                                                     01340
         crlf();                                                       01341
         typeas(&astr); %let operator see what is happenning%          01342
         END                                                           01343
      ELSE                                                             01344
         BEGIN %Garden variety document%                               01345
         lastfnum _ 0;                                                 01346
         poutput(stid, $docjfn, $docwfn);                              01347
         chklpt();                                                     01348
         copylpt(TRUE, docjfn, 0, -1);                                 01349
         RETURN                                                        01350
         END;
```

```
                                                                    01351
   %Output header%                                                  01352
      poutput(stid, $hdrjfn, $hdrwfn);                              01353
   %If new file, get rest of document%                              01354
      IF (lastfnum := fnum) # fnum THEN                             01355
         BEGIN %New document%                                       01356
         IF NOT FIND SF(stid) C".PSW=0;.HLT;"] ^z2 < C"P."] ^z1 >
         THEN                                                       01357
            SIGNAL(2, $"Bad print file");                           01358
         ST z1 z2 _ NULL; %delete IRS and HALT%                     01359
         %output entire file%                                       01360
            poutput(stid, $docjfn, $docwfn);                        01361
         %Now get start of document from header file%               01362
            r1 _ hdrjfn;                                            01363
            IF NOT SKIP !JSYS sizef THEN                            01364
               err($"Print File Error");                            01365
            docstrt _ r2-1;                                         01366
         END;                                                       01367
   %Now copy to lpt:%                                               01368
      chklpt();                                                     01369
      copylpt(TRUE, hdrjfn, 0, -1); %header%                        01370
      copylpt(FALSE, docjfn, docstrt, -1); %document%               01371
   RETURN;                                                          01372
   END.
                                                                    01373
(poutput)PROC(stid, jfn, jfnnam);                                   01374
   LOCAL opjfn;                                                     01375
   REF jfn, jfnnam;                                                 01376
   %Close file if it is open%                                       01377
      IF jfn # 0 THEN  sysclose(jfn := 0);                          01378
   %now call processor%                                             01379
      tda.dacsp _ stid;                                             01380
      coutproc (&jfnnam, &tda, opprdv, 0 %flags for output processor
      %);                                                           01381
   %Now re-open file%                                               01382
      jfn _ sgtjfn(getgtjflg(write, 0, oldvrsn), &jfnnam, $lit);
                                                                    01383
      IF NOT sysopen(jfn, readwrite, chrtyp, $lit) THEN             01384
         BEGIN                                                      01385
         %May have failed for timing reasons%                       01386
         r1 _ 1000;                                                 01387
         !JSYS disms;                                               01388
         IF NOT sysopen(jfn, readwrite, chrtyp, $lit) THEN          01389
            err($lit);                                              01390
         END;                                                       01391
   RETURN END.
                                                                    01392
(nicsitecopies)PROC(dwstid, hcdstid);                               01393
   RETURN;                                                          01394
   END.
                                                                    01395
(getdauth)PROC(stid, authsr);                                       01396
   %Set up astr authsr to have the list of authors of document
   indicated by distfile entry stid.  Append address of first author
   to end of string%                                                01397
   LOCAL idfno;                                                     01398
```

```
   LOCAL STRING tempsr[30], tempadsr[250], tempauthor[50];
   LOCAL TEXT POINTER z1, z2;
   REF authsr;
   idfno _ 0;
   ON SIGNAL ELSE
      BEGIN
      sigclose(idfno := 0);
      IF sysgnl # -4 THEN SIGNAL(1);
      END;
   IF NOT FIND SF(stid) ["Author(s): "] ['/] $NP ^z1 ('&/'^/) ^z2
   THEN
      BEGIN
      IF NOT FIND SF(stid) [".HJOURNAL="] [NP 1$2D '- 3L] < [1$NP]
      ^z2 ['" / ';] > CH ^z1 THEN SIGNAL(1, $"Bad Distfile Entry");

      *tempauthor* _ z1 z2, '; ;
      FIND SF(*tempauthor*) ^z1 ('&/'^/) ^z2;
      END;
   *authsr* _ NULL;
   idfno _ open(0, jflname($"identfile"));
   %Get the names of the authors%
      intids(0);
      WHILE getids($z1, &authsr, 1, idfno) DO
         *authsr* _ *authsr*, EOL;
   %Now get the addresss of the frst author%
      IF (authsr.L = 0) OR NOT FIND z2 L $(LD / '- / '') ^z1 THEN
      err($"Illegal Author Field");
      *tempsr* _ z2 z1;
      IF NOT ckident($tempsr, $xlit, idfno) THEN err($"Illegal
      Author");
   %Now extract address and put it all together%
      laddress($xlit, $tempadsr, 0, 0, idfno);
      close(idfno := 0);
      *authsr* _ *authsr*, *tempadsr*;
   RETURN;
   END.

(getdwork)PROC(hcdstid, authsr);
   %Open and set up dwork file for printing.
      Return stid of dwork file if ok, zero if not.
      %
   REF authsr;
   LOCAL dwstid, docstid;
   LOCAL TEXT POINTER z1, z2, z3, t1;
   dwstid _ docstid _ 0;
   ON SIGNAL ELSE
      BEGIN
      close(dwstid.stfile := 0);
      close(docstid.stfile := 0);
      END;
   %Open dwork, and set up the origin statement%
      dwstid _ openlock(0, jflname($"distwork"));
      enablaccess(dwstid.stfile, jrnlaccess);
      resetf(dwstid.stfile); % reset file %
      FIND SF(hcdstid) ['')] ^z1 ( $NP '( ['')] / ) ^z2; %delimit link%
```

```
      %TEMPORARY kludge for output processor glitch when no title
      (.H1) directive%                                                01448
         IF (FIND z2 [".HJOURNAL="] CH ['" ';] $NP ^t1) AND NOT (FIND
         "Title:") THEN ST t1 t1 _ "Title: .H1=", '", ' , '", "; ";
                                                                      01449
      FIND SE(z2) ((CEOL EOL] $EOL > 2EOL) / ) > ^z3; %position z3 at
      end, but before "Master Copy Printed", etc.%                    01450
      FIND z3 > (".PEL;" ((CEOL ("Master" / "Access" / "Engelbart")]
      < CEOL] >) / SE(z3)) ^z3); %move past comment if here%          01451
      ST dwstid _ *authsr*, ".LMS=20;
      .GCR=3; To: .LMS=24;

      .PEL;
      .LMS = 0;", SF(hcdstid) z1, z2 z3;                              01452
   %Now set up the rest of the file%                                  01453
      IF docstid _ jdelloaddoc(hcdstid) THEN                          01454
         BEGIN                                                        01455
         ccopgro(dwstid, levsuc, docstid _ getsub(docstid),
         getail(docstid), FALSE, 0);                                  01456
         close(docstid.stfile := 0);                                  01457
         END;                                                         01458
   RETURN(dwstid);                                                    01459
   END.
                                                                      01460
(getjnm)PROC(ptr, numsr);                                             01461
   %Get journal number field from distfile entry %                    01462
   LOCAL TEXT POINTER z1, z2;                                         01463
   REF ptr, numsr;                                                    01464
   IF NOT (FIND ptr > C'(] ('J/'j) ^z1 4$5D ^z2 ) THEN                01465
      err($"Bad distfile header entry");                              01466
   *numsr* _ z1 z2;                                                   01467
   RETURN;                                                            01468
   END.
                                                                      01469
(jdelloaddoc)PROC(stid);                                              01470
   %assume that stid points to a distfile entry, and get the branch
   containing the coresponding document, returning stid of document
   or zero if it is hard copy%                                        01471
   LOCAL docstid, adstr[40];                                          01472
   LOCAL TEXT POINTER z1;                                             01473
   LOCAL STRING user[20], filename[50], statname[30], vsp[15];        01474
   docstid _ 0;                                                       01475
   ON SIGNAL ELSE                                                     01476
      BEGIN                                                           01477
      close(docstid.stfile := 0);                                     01478
      SIGNAL(1);                                                      01479
      END;                                                            01480
   IF FIND SF(stid) [')] ^z1  ["Hard Copy--Location"] THEN RETURN(0);
                                                                      01481
   %Now parse link---z1 points to spaces before it%                   01482
      lnkpspc(1, $z1, $user, $filename, $statname, $vsp, $adstr);
                                                                      01483
      *filename* _ *filename*, ".NLS";                                01484
   %Now open file and find proper branch%                             01485
      docstid.stpsid _ origin;                                        01486
      docstid.stfile _ open(0, $filename);                            01487
```

```
      IF *statname*[1] IN ['0, '9] THEN RETURN(docstid); %a terrible
      way of checking if it is a document or message---change after
      backlog is gone %                                              01488
      makeptr(docstid, $z1);                                         01489
      specreg($statname, nametyp, $z1);                              01490
      IF z1 = endfile THEN                                           01491
         BEGIN                                                       01492
         *lit* _ *filename*, " Branch ", *statname*, " Not Found";
                                                                     01493
         err($lit);                                                  01494
         END;                                                        01495
      RETURN(z1);                                                    01496
   END.
                                                                     01497
(hcstatus)PROC;                                                      01498
   LOCAL hcdstid, stid1, stid2, distcount, collcount, niccount,
   count, idjfn;                                                     01499
   LOCAL TEXT POINTER z1, z2;                                        01500
   %return numbet of expedited printouts to be done, no. of normal
   docs%                                                             01501
   hcdstid _ 0;                                                      01502
   ON SIGNAL ELSE                                                    01503
      BEGIN                                                          01504
      sigclose(hcdstid.stfile := 0);                                 01505
      sigclose(idjfn := 0);                                          01506
      END;                                                           01507
   hcdstid.stpsid _ origin;                                          01508
   hcdstid.stfile _ open(0, jflname($"hcdistfile"));                 01509
   idjfn _ open(0, jflname($"identfile"));                           01510
   stid1 _ getsub(hcdstid);                                          01511
   distcount _ collcount _ niccount _ 0;                             01512
   WHILE stid1 # hcdstid DO                                          01513
      BEGIN                                                          01514
      stid2 _  getsub(stid1);                                        01515
      WHILE stid2 # stid1 DO                                         01516
         BEGIN                                                       01517
         distcount _ distcount + phcopy(stid2);                      01518
         stid2 _ getsuc(stid2);                                      01519
         END;                                                        01520
      collcount _ collcount +                                        01521
         (NOT FIND SF(stid1) ["Master Copy Printed"]);               01522
      niccount _ niccount + ((FIND SF(stid1) ["Sub-Collections: "]
      ^z1 [';] ^z2 BETWEEN z1 z2(["NIC"])) AND NOT FIND SF(stid1)
      ["NIC Copy Printed"]);                                         01523
      stid1 _ getsuc(stid1);                                         01524
      END;                                                           01525
   close(idjfn := 0);                                                01526
   close(hcdstid.stfile := 0);                                       01527
   RETURN(distcount, collcount, niccount);                           01528
   END.
                                                                     01529
(jdquit)PROCEDURE;                                                   01530
   %Print out hcdistfile if printing has been done, and garbage
   collect distribution file%                                        01531
   LOCAL hcdstid, svspc, svstd;                                      01532
   hcdstid _ 0;                                                      01533
```

```
   ON SIGNAL ELSE sigclose(hcdstid.stfile := 0);                    01534
   IF prntfg  THEN                                                  01535
      BEGIN                                                         01536
      prntfg _ FALSE;                                               01537
      % print out distfile%                                         01538
         % save viewspecs and stid fr they will be changed by outptr
         %                                                          01539
            svspc _ tda.davspec;                                    01540
            svstd _ tda.dacsp;                                      01541
         outptr(IF jdebug THEN $"<Duval1>hcdistfile" ELSE
         $"<Journal>hcdistfile", &tda, 1, TRUE);                    01542
         % restore dacsp and davspec %                             01543
            tda.davspec _ svspc;                                    01544
            tda.dacsp _ svstd;                                      01545
      %garbage collect hcdistfile%                                 01546
         hcdstid _ openlock(0, jflname($"hcdistfile"));             01547
         gdistf(hcdstid:=0);                                        01548
       END;                                                         01549
   RETURN;                                                          01550
   END.

                                                                    01551
(lptclose)PROC;                                                     01552
   %Close out all of the lpt machinery%                             01553
   IF lptjfn THEN sysclose(lptjfn := 0);                            01554
   IF docjfn THEN sysclose(docjfn := 0);                            01555
   IF hdrjfn THEN sysclose(hdrjfn := 0);                            01556
   %Global stids and  file variables%                               01557
      docstrt _ lastfnum _ lptused _ idfile _ 0;                    01558
   RETURN;                                                          01559
   END.

                                                                    01560
(copylpt)PROC(resflg, jfn, startb, stopb);                          01561
   LOCAL count;                                                     01562
   %First set file pointer to start%                                01563
      r1 _ jfn; r2 _ startb; IF NOT SKIP !JSYS sfptr THEN err($"I/O
      JSYS error");                                                 01564
      %Now set eof%                                                 01565
         IF stopb = -1 THEN                                         01566
            BEGIN                                                   01567
            %get size of file%                                      01568
            r1 _ jfn; IF NOT SKIP !JSYS sizef THEN err($"I/O JSYS
            error");                                                01569
            stopb _ r2;                                             01570
            END                                                     01571
         ELSE                                                       01572
            BEGIN                                                   01573
            %set size of file%                                      01574
            r1.RH _ jfn; r1.LH _ 12B; r2 _ -1; r3 _ stopb;          01575
            !JSYS chfdb;                                            01576
            END;                                                    01577
      %Send out restore if necessary%                               01578
         %output processor paginates for us now%                    01579
      %Get count and output%                                        01580
         count _ stopb - startb;                                    01581
         lptused _ lptused + count;                                 01582
         WHILE count > 0 DO                                         01583
```

```
            BEGIN                                                       01584
            %get string from input%                                     01585
               count _ count - 2560;                                    01586
               r3 _ -(IF count < 0 THEN 2560+count ELSE 2560);          01587
               r1 _ jfn; r2 _ chbptr(0) + $ckpag;                       01588
               !JSYS sin; %read string from input file%                 01589
            %copy to lpt%                                               01590
               r1 _ lptjfn; r2 _ chbptr(0) + $ckpag;                    01591
               r3 _ -(IF count < 0 THEN 2560+count ELSE 2560);          01592
               !JSYS sout;                                              01593
            END;                                                        01594
      RETURN                                                            01595
      END.
                                                                        01596
(chklpt)PROC;                                                           01597
   LOCAL cntr;                                                          01598
   IF lptjfn # 0 THEN BEGIN                                             01599
      IF lptused < 75000 THEN RETURN ELSE sysclose(lptjfn:=0);          01600
      END;                                                              01601
   lptused _ 1;                                                         01602
   IF (lptjfn _ sgtjfn(getgtjflg(write, 0, 0), $ptstr, $lit)) = 0
   THEN err($lit);                                                      01603
   FOR cntr _ 0 UP 1 UNTIL > 90 DO                                      01604
      IF sysopen(lptjfn, write, lpttype, $lit) THEN RETURN              01605
      ELSE pause(20000);                                                01606
   lptjfn _ lptused _ 0;                                                01607
   err($"Printer busy too long");                                       01608
   END.
                                                                        01609
(tagjdel)PROC(stid, astr);                                              01610
   REF astr;                                                            01611
   %Tag the statement stid with string + "Printed By " OPERATOR"%
                                                                        01612
   ST stid _ SF(stid) SE(stid), EOL, *astr*, " Printed by ", *opstr*;
                                                                        01613
   RETURN;                                                              01614
   END.
                                                                        01615
% Build Distribution file enties %                                      01616
   (dsecdoc) PROC(number, nethcflag, usetejlink, jcstid, idnum);
   %process secondary distribution request in jworkfile into distfile%
                                                                        01617
   LOCAL jxcstid;                                                       01618
   LOCAL TEXT POINTER z1, z2, z3;                                       01619
   LOCAL STRING notstr[55];                                             01620
   REF number;                                                          01621
   jxcstid _ 0;                                                         01622
   ON SIGNAL ELSE                                                       01623
      BEGIN                                                             01624
      close(jxcstid.stfile :=0);                                        01625
      RETURN (FALSE);                                                   01626
      END;                                                              01627
   FIND jwp1 > ([("(J" / "(j"] ^z1) _2z1 SE(jworkstid) > ^z3;           01628
   ST jworkstid _ z1 z3, EOL, "Secondary Distribution Copy";            01629
   IF FIND jwp1 ["(Expedite)"] ^z3 < ["("] ^z2 >  THEN ST z2 z3 _
   NULL;                                                                01630
```

```
   z2 _ getsub(jworkstid);                                          01631
   z1 _ getsuc(z2);                                                 01632
   FIND jwp1 [*)] ^z3;                                              01633
   IF NOT usetejlink THEN %must get link from catalog%              01634
      BEGIN                                                         01635
      IF NOT (z2 _ gtcatent (&number, 0, jcstid.stfile, FALSE)) AND
      NOT (z2 _ jxcstid _ gtcatent (&number, 0, 0, FALSE)) THEN err
      ($"No such document");                                        01636
      z2 _ getsub(z2);                                              01637
      END;                                                          01638
   IF FIND SF(z2) ["Link"] [*(] ^z2 _z2 THEN                        01639
      BEGIN                                                         01640
      ST z3 z3 _ SP, z2 SE(z2); %link%                              01641
      END;                                                          01642
   %make notstr%                                                    01643
      z2 _ getsuc(z1);                                              01644
      *notstr* _ "(Secondary Distribution Copy from ", SF(z2) SE(z2),
      *);                                                           01645
   %set up ident list pointer%                                      01646
      FIND SF(z1) ^z1;                                              01647
   diststid _ ccopsta(diststid, levsuc, jworkstid, 0, 0);           01648
   distd1(diststid, $z1, idnum, $notstr, nethcflag);                01649
   close (jxcstid.stfile:=0);                                       01650
   RETURN(TRUE); END.
                                                                    01651
(distdoc)PROC(astr, linkuser,  number, mesflg, headstr, idnum,
nethcflag, recordedflg);                                            01652
   %distribute document identified by jworkstid%                    01653
   %do net and hc delivery if nethcflag%                            01654
   LOCAL stid;                                                      01655
   LOCAL TEXT POINTER z1, z2, z3, z4, z5;                           01656
   REF astr, number, linkuser, headstr;                             01657
   %astr contains file name upon call, Return link to document in
   astr%                                                            01658
   %ton 9%                                                          01659
   diststid _  ccopsta(diststid, levsuc, jworkstid, FALSE, 0);      01660
   FIND SF(diststid) ^z1 [*,] [";;;;"] ^z2 _3z2;                    01661
   ST z1 z2 _ *headstr*;                                            01662
   CASE recordedflg OF                                              01663
      = TRUE:                                                       01664
         BEGIN                                                      01665
         CASE mesflg OF                                             01666
            =0:  *astr* _ *(, *linkuser*, ", ", *astr*,  ", 1:w)";
                                                                    01667
            =1: *astr* _ *(, *linkuser*, ", ", *astr*, %journal file
            name%                                                   01668
               ", J", *number*, ":gw) ";                            01669
            =hcopyv:  *astr* _ "(Hard Copy--",  *linkuser*, ", ",
            *astr*,  ", )";                                         01670
            ENDCASE snkerr($"Illegal Message Type");                01671
         END;                                                       01672
      ENDCASE                                                       01673
         BEGIN % unrecorded: resides in jworkstid.stfile %          01674
         % make branch name start with "U" instead of "J" %         01675
            IF FIND > z1 ["(J"] ^z3 ^z4 _z4 THEN                    01676
               ST z1 _ z1 z4, *U, z3 SE(z1);                        01677
```

```
        % make link to actual location, a file with an "NLU"
        extension %                                                    01678
           FIND SF(*astr*) ^z5;                                        01679
           FIND z5 ['<] ^z5 ['>] ^z3 ^z2 _z2;                          01680
           *astr* _ '(, z5 z2, ', , z3 SE(*astr*),  ", :1 )";          01681
        END;                                                           01682
   IF mesflg # hcopyv THEN                                             01683
      BEGIN                                                            01684
      FIND SF(diststid) ['')]  ^z1;                                    01685
      ST z1 z1 _ "  ", *astr*; %Insert link into header for
      distribution%                                                    01686
      END;                                                             01687
   %now insert individuals on distribution list%                       01688
      IF FIND SF(diststid)["Distribution: "] ['/] ^ z1 THEN            01689
         distd1(diststid, $z1, idnum, IF mesflg = hcopyv THEN $"Hard
         Copy Document" ELSE 0, nethcflag);                            01690
   %now provide for author copies%                                     01691
      IF FIND SF(diststid) ["Author(s):"] ['/] ^ z1 THEN               01692
         distd1(diststid, $z1, idnum, $"Author Copy", nethcflag)
                                                                       01693
      ELSE snkerr($"System Error (No Author Field)");                  01694
   %toff 9%                                                            01695
   RETURN END.
                                                                       01696
(distd1)PROC(dstid, ptr, idnum, note, nethcflag);                      01697
   %This procedure does the actual manipulation of the distribution
   file to put entries in for the individual reciepients.              01698
      dstid is te stid of the branch head in the distribution fule
                                                                       01699
      ptr is a pointer to the distribution list                        01700
      idnum is the identification file number (if non-zero)            01701
      note is optionally a string which will be appended to the entry
      in the comments/notes field (in addition to any comments which
      may be in the distribution list).  Don't do any net or hc
      delivery if nethcflag is FALSE.                                  01702
   %                                                                   01703
   LOCAL idfnum, stid, ldelflags, netflag, hcflag;                     01704
   LOCAL TEXT POINTER z1, z2, z3;                                      01705
   LOCAL STRING ddstr[30];                                             01706
   REF ptr, note;                                                      01707
   netflag _ nethcflag.delnet;                                         01708
   hcflag _ nethcflag.delhc;                                           01709
   IF idnum = 0 THEN                                                   01710
      BEGIN                                                            01711
      idfnum _ 0;                                                      01712
      ON SIGNAL ELSE sigclose(idfnum := 0);                            01713
      idfnum _ open(0, jflname($"identfile"));                         01714
      END                                                              01715
   ELSE idfnum _ idnum;                                                01716
   intids(0);                                                          01717
   LOOP                                                                01718
      BEGIN                                                            01719
      *lit* _ NULL;                                                    01720
      IF NOT getids(&ptr, $lit, 0, idfnum) THEN EXIT;                  01721
      ldelflags _ ldelivery(asrref($lit));                             01722
      *ddstr* _ NULL;                                                  01723
```

```
      IF netflag AND ldelflags.delnet THEN                          01724
         *ddstr* _ *ddstr*, " do-net";                              01725
      IF hcflag AND ldelflags.delhc THEN                            01726
         *ddstr* _ *ddstr*, " do-hc";                               01727
      IF ldelflags.delol THEN                                       01728
         *ddstr* _ *ddstr*, " do-ol";                               01729
      IF ddstr.L = 0 THEN REPEAT LOOP;                              01730
      getiid($lit, 0, $z1, $z2);                                    01731
      *lit* _ '(, z1 z2, '); %Ident%                                01732
      IF gbotids($z3) THEN                                          01733
         FIND z3 ELSE FIND ptr;                                     01734
      IF FIND < CH > ') THEN % comment field present %              01735
         BEGIN                                                      01736
         IF NOT FIND < CH ^z2  [']'[] >  CH ^z1 THEN                01737
            snkerr($"System Error (Bad distribution List)");        01738
         % change do-ol by adding 'a (action) %                     01739
            IF ldelflags.delol THEN                                 01740
               IF FIND " [ ACTION ]" THEN *ddstr* _ *ddstr*, 'a;
                                                                    01741
               % format must aggree with that in (nls, csendmail,
               convjdist) %                                         01742
            IF &note THEN                                           01743
               *lit* _ *lit*, EOL, "Note: ", z1 z2, EOL, *note*, ";;"
                                                                    01744
            ELSE                                                    01745
               *lit* _ *lit*, EOL, "Note: ", z1 z2, ";;";           01746
            END                                                     01747
      ELSE                                                          01748
         IF &note THEN *lit* _ *lit*, EOL, "Note: ", *note*, ";;";
                                                                    01749
      *lit* _ *lit*, " *DEL:", *ddstr*;                             01750
      cis(dstid, $lit, $downstr);                                   01751
      END;                                                          01752
   IF NOT idnum THEN close(idfnum);                                 01753
   RETURN END.
                                                                    01754
% Distribution file manipulation-- garbage collection, updating %  01755
  (gdistf)PROC(hcdstid);                                            01756
   %Garbage collect hard copy distribution file%                    01757
   LOCAL stid, stid1, idjfn;                                        01758
   LOCAL TEXT POINTER z1,z2;                                        01759
   LOCAL STRING tempsr[50];                                         01760
   %Set up  subsy name%                                             01761
      r1 _ getsbn($"GCOLDF"); !JSYS setnm;                          01762
   idjfn _ 0;                                                       01763
   ON SIGNAL ELSE                                                   01764
      BEGIN                                                         01765
      sigclose(idjfn := 0);                                         01766
      sigclose(hcdstid.stfile := 0);                                01767
      END;                                                          01768
   %open hcdist file%                                               01769
      IF NOT hcdstid.stfile THEN hcdstid _ openlock(0,
      jflname($"hcdistfile"));                                      01770
      idjfn _ open(0, jflname($"identfile"));                       01771
   %loop through file looking for stuff to delete%                  01772
      stid _ getsub(hcdstid);                                       01773
```

```
      WHILE stid # hcdstid DO                                              01774
         BEGIN                                                             01775
         stid1 _ getsub(stid);                                             01776
         WHILE stid1 # stid DO                                             01777
            BEGIN                                                          01778
            CCPOS SF(stid1);                                               01779
            xtrnam($tempsr, $swork, -1);                                   01780
            ckident($tempsr, $xlit, idjfn);                                01781
            IF NOT (phcopy(stid1) OR donline(stid1) OR
            dnetdel(stid1)) THEN                                           01782
               cdelsta(stid1 := getsuc(stid1), FALSE, 0)                   01783
            ELSE stid1 _ getsuc(stid1);                                    01784
            END;                                                           01785
         IF getsub(stid) = stid THEN                                       01786
            BEGIN %all copies printed%                                     01787
            CCPOS SF(stid);                                                01788
            IF                                                             01789
               NOT mastacprintflag                                         01790
               OR (                                                        01791
                  (FIND ^z1 [^"Master Copy Printed"])                      01792
                  %no NIC copies for now%                                  01793
                     %AND NOT((FIND z1 [^"Sub-Collections: "] ^z1 [^";]
                     ^z2 BETWEEN z1 z2 ([^"NIC"])) AND NOT FIND SF(z1)
                     [^"NIC Copy Printed"])%                               01794
                  )                                                        01795
               OR (FIND z1 [EOL "Secondary Distribution Copy"])            01796
               OR getcacc (stid, 0, 0, 0) %private document%               01797
               THEN cdelsta(stid := getsuc(stid), FALSE, 0)                01798
            ELSE stid _ getsuc(stid);                                      01799
            END                                                            01800
         ELSE stid _ getsuc(stid);                                         01801
         END;                                                              01802
   %Update and exit%                                                       01803
      closeu(hcdstid.stfile := 0);                                         01804
      close(idjfn);                                                        01805
      r1 _ getsbn($"HCJDEL");                                              01806
      !JSYS setnm;                                                         01807
      RETURN                                                               01808
      END.
                                                                           01809
(updhcdist)PROC;                                                           01810
   %Update DISTFILE from TDISTFILE, and return TRUE if there is
   anything in the distribution file%                                      01811
   LOCAL ldiststid, hcdstid, retval, stid, stidd, deldel;                  01812
   LOCAL STRING adsr[45];                                                  01813
   %Initialise LOCALS%                                                     01814
      hcdstid _ 0;                                                         01815
      hcdstid.stpsid _ origin;                                             01816
      ldiststid _ hcdstid;                                                 01817
   %Set Signal to close files%                                             01818
      ON SIGNAL ELSE                                                       01819
         BEGIN                                                             01820
         sigclose(hcdstid.stfile);                                         01821
         sigclose(ldiststid.stfile);                                       01822
         END;                                                              01823
   ldiststid _ openlock(0, jflname($"distfile"));                          01824
```

```
   IF (getsub(ldiststid)).stpsid = origin THEN                          01825
      BEGIN                                                             01826
      close(ldiststid.stfile);                                          01827
      RETURN(TRUE); %Assume that there will always be something in
      the dist file%                                                    01828
      END;                                                              01829
   hcdstid _ openlock(0,jflname($"hcdistfile"));                        01830
   enablaccess(hcdstid.stfile, jrnlaccess);                             01831
   enablaccess(ldiststid.stfile, jrnlaccess);                           01832
   cmovgro(stid_getail(getsub(hcdstid)), levsuc, ldiststid _
   gethed(getsub(ldiststid)), getail(ldiststid), FALSE, 0);             01833
   closeu(idiststid.stfile := 0);                                       01834
   UNTIL (stid _ getsuc(stid)).stpsid = origin DO                       01835
      BEGIN                                                             01836
      *xlit* _ SP;                                                      01837
      stidd _ getsub (stid);                                            01838
      UNTIL stidd = stid DO BEGIN                                       01839
         CCPOS SF(stidd);                                               01840
         xtrnam ($adsr, $swork, -1);                                    01841
         IF adsr.L = 0 OR FIND SF(*xlit*) [SP *adsr* SP]                01842
            THEN deldel _ stidd                                         01843
            ELSE BEGIN                                                  01844
               *xlit* _ *xlit*, *adsr*, SP;                             01845
               deldel _ (IF FIND [EOL] ("Note: *Exclude") THEN stidd
               ELSE FALSE);                                             01846
               END;                                                     01847
         stidd _ getsuc(stidd);                                         01848
         IF deldel THEN cdelgro(deldel, deldel, FALSE,0);               01849
         END;                                                           01850
      END;                                                              01851
   retval _ (getsub(hcdstid)).stpsid # origin;                          01852
   closeu(hcdstid.stfile := 0); %update file%                           01853
   RETURN(retval);                                                      01854
   END.

                                                                        01855
% Message files %                                                       01856
   (entrjdoc)PROC(stid, number, headstr);                               01857
      %Enter journal documnnt into active file as identified by stid%
                                                                        01858
      LOCAL TEXT POINTER z1, z2;                                        01859
      REF number,headstr;                                               01860
      %ton 5%                                                           01861
      %first copy over document%                                        01862
         stid _ getail(getsub(stid));                                   01863
         stid_ ccopgro(stid, levsuc, jworkstid, jworkstid, FALSE, 0);
                                                                        01864
         FIND SF(stid) ^z1 ["">"] ["","] [";] ^z2;                      01865
         ST z1 z2 _ *headstr*;                                          01866
         stid.stpsid _ origin;                                          01867
         ST SF(stid) _ SF(stid) SE(stid), " ", *number*; %update header%
                                                                        01868
      %toff 5%                                                          01869
      RETURN END.

                                                                        01870
   (findactive)PROC(numsmt, jcstid);                                    01871
      %Find the actve journal file, given the required number of
```

```
   statements in numsmt, and jcstid is the stid of the origin
   statement of the journal control file.                          01872
   If there is no active journal file, or there is not enough room in
   the active journal file, then close it, and create a new active %
                                                                   01873
   LOCAL usedsmt, stid;                                            01874
   LOCAL TEXT POINTER z1, z2;                                      01875
   LOCAL STRING tempsr[15];                                        01876
   %ton 4%                                                         01877
   %First look for active%                                         01878
      IF (stid _ namelook(jcstid,$"ACTIVE"))= endfil THEN
      RETURN(newactive(jcstid), numsmt);%get new active file% %need
      new active file%                                             01879
   %now check room here%                                           01880
      %first get current number of statements in file%             01881
         FIND SF(stid) C"Used = "] $NP ^z1 $D ^z2;                 01882
         *tempsr* _  z1 z2; %number of used statements%            01883
         usedsmt _ VALUE($tempsr);                                 01884
      %now check to see if room%                                   01885
         IF usedsmt+numsmt > jmaxsmt THEN                          01886
            BEGIN %no more room in this active%                    01887
            relactive(stid, jcstid); %release it%                  01888
            RETURN(newactive(jcstid), numsmt);                     01889
            END                                                    01890
         ELSE RETURN(openactive(stid, FALSE), usedsmt+numsmt);     01891
   END.
                                                                   01892
(newactive)PROC(jcstid);                                           01893
   %This sets up a new active file, and opens it, returning the stid
   of the origin statement%                                        01894
   LOCAL STRING tempsr[35];                                        01895
   LOCAL TEXT POINTER tp1, tp2;                                    01896
   IF (jcstid := getail(getsub(jcstid))) = jcstid  THEN            01897
      *tempsr* _  "(ACTIVE) (JRNL1) Used = 0"                      01898
   ELSE                                                            01899
      BEGIN                                                        01900
      CCPOS SF(jcstid);                                            01901
      IF NOT FIND $NP "(JRNL" ^tp1 1$D ^tp2 THEN                   01902
         SIGNAL(-5, $"Bad Entry in Jcat File"); %Make it look like a
         bad file%                                                 01903
      *tempsr* _ tp1 tp2;                                          01904
      bumpstr($tempsr);                                            01905
      *tempsr* _ "(ACTIVE) (JRNL", *tempsr*, ") Used = 0";         01906
      END;                                                         01907
   jcstid _ cis(jcstid, $tempsr, $sucdir);                         01908
   csetnsta(jcstid, '(, ') );                                      01909
   RETURN(openactive(jcstid, TRUE));                               01910
   END.
                                                                   01911
(openactive)PROC(stid, newfile);                                   01912
   %This routine fetches the active name from the statement
   identified by stid, and opens the file, returning the stid of the
   origin statement%                                               01913
   %Note that lock is not a concern because JCAT MUST BE OPEN which
   is locked%                                                      01914
   LOCAL STRING tempsr[15];                                        01915
```

```
    LOCAL TEXT POINTER tp1, tp2;                                        01916
    CCPOS SF(stid);                                                     01917
    IF NOT FIND $NP "(ACTIVE) (" ^tp1 [*)] ^tp2 _tp2 THEN               01918
       err($"Jctl Error");                                              01919
    *tempsr* _ tp1 tp2;                                                 01920
    IF newfile THEN                                                     01921
       BEGIN                                                            01922
       stid _ openwk(0, jflname($tempsr));                              01923
       setaccess(stid.stfile, jrnlaccess);                              01924
       closeu(stid.stfile);                                             01925
       END;                                                             01926
    stid _ 0;                                                           01927
    stid _ openlock(0, jflname($tempsr));                               01928
    %toff 4%                                                            01929
    RETURN(stid);                                                       01930
    END.
                                                                        01931
(relactive)PROC(stid);                                                  01932
    %This procedure releases the active identified in the jcat by
    stid%                                                               01933
    LOCAL TEXT POINTER tp1;                                             01934
    CCPOS SF(stid);                                                     01935
    FIND ["(ACTIVE) "] ^tp1;                                            01936
    ST tp1 _ tp1 SE(tp1);                                               01937
    RETURN END.
                                                                        01938
% Read fields from distribution file entry. %                           01939
(phcopy)PROC(stid);                                                     01940
    %Return true if hard copy needs to be printed for statement
    identified by stid%                                                 01941
    IF FIND SE(stid)< [*"] > "*DEL:" [" do-hc"] THEN RETURN(TRUE) ELSE
    RETURN(FALSE);                                                      01942
    END.
                                                                        01943
(donline)PROC(stid);                                                    01944
    %Return (1) true if on line distribution should be done for
    statement identified by stid, and (2) true if ACTION type send.%
                                                                        01945
    LOCAL retflg;                                                       01946
    retflg _ FALSE;                                                     01947
    IF FIND SE(stid) [*"] > "*DEL:" [" do-ol"] THEN                     01948
       BEGIN                                                            01949
       IF FIND 'a THEN retflg _ TRUE;                                   01950
       RETURN(TRUE, retflg);                                            01951
       END                                                              01952
    ELSE RETURN(FALSE, FALSE);                                          01953
    END.
                                                                        01954
(dnetdel)PROC(stid);                                                    01955
    %Return true if net distribution should be done for statement
    identified by stid.%                                                01956
    IF FIND SE(stid)< [*"] > "*DEL:" [" do-net"] THEN RETURN(TRUE)
    ELSE RETURN(FALSE);                                                 01957
    END.
                                                                        01958
% Online delivery citation format %                                     01959
```

```
(olhed) PROC (adrstid, mflag, stid, l1, l2, recorded, wrkstr);  01960
   %Format stid to online delivery citation format%             01961
   %l1, l2 delimit location of document or message (link)%      01962
   %leaves citation in lit and the comment, if any, in lit2%    01963
   %mflag = TRUE for message; recorded = TRUE if catalogued/archived
   %                                                            01964
   %Note: message text will appear as separate plex below citation. %
                                                                01965
   LOCAL TEXT POINTER z1;                                       01966
   REF l1, l2, wrkstr;                                          01967
   ON SIGNAL ELSE                                               01968
      IF mflag THEN RETURN(FALSE);                              01969
   CCPOS SF(stid);                                              01970
   FIND ^z1;                                                    01971
   *lit* _ *[hjournal($z1, &wrkstr)]*,                          01972
      EOL, *[jtitle($z1, &wrkstr)]*,                            01973
      EOL, *[hcopyl($z1, &wrkstr)]*,                            01974
      *[olloc(mflag, recorded, &l1, &l2, &wrkstr)]*,            01975
      EOL, *[jnote(adrstid, &wrkstr, 0)]*;                      01976
   % set up text pointer for getjcomment. %                     01977
      jwp1 _ z1; jwp1[1] _ z1[1];                               01978
   IF [getjcomment(&wrkstr)].L <= 0 THEN                        01979
      *wrkstr* _ NULL                                           01980
   ELSE                                                         01981
      *wrkstr* _ "Comments: ", *wrkstr*;                        01982
   RETURN(TRUE);                                                01983
   END.
                                                                01984
(hjournal)PROC(z1, wrkstr);                                     01985
   LOCAL TEXT POINTER z2 , z3, z4, z5;                          01986
   REF z1, wrkstr;                                              01987
   CCPOS z1;                                                    01988
   IF FIND [".HJOURNAL="] ['"]  ^z2 ['"] ^z3 _z3 THEN           01989
      BEGIN                                                     01990
      IF FIND BETWEEN z2 z3([".SPLIT;"] ^z4 < ['.] ^z5 >) THEN  01991
         *wrkstr* _ z4 z3, " [", z2 z5, ']                      01992
      ELSE *wrkstr* _ z2 z3                                     01993
      END                                                       01994
   ELSE *wrkstr* _ NULL;                                        01995
   RETURN(&wrkstr)                                              01996
   END.
                                                                01997
(jtitle)PROC(z1, wrkstr);                                       01998
   LOCAL TEXT POINTER z2 , z3;                                  01999
   REF z1, wrkstr;                                              02000
   CCPOS z1;                                                    02001
   IF FIND ["Title: "] ['"] ^z2 ['"] ^z3 _z3 THEN               02002
      *wrkstr* _ z2 z3                                          02003
   ELSE                                                         02004
      *wrkstr* _ NULL;                                          02005
   RETURN(&wrkstr)                                              02006
   END.
                                                                02007
(hcopyl)PROC(z1, wrkstr);                                       02008
   REF z1, wrkstr;                                              02009
   CCPOS z1;                                                    02010
```

```
   IF FIND [*Hard Copy--Location: *] THEN                              02011
      *wrkstr* _ "Hard Copy Only--"                                    02012
   ELSE                                                                02013
      *wrkstr* _ NULL;                                                 02014
   RETURN(&wrkstr)                                                     02015
   END.
                                                                       02016
(olloc)PROC(mflag, recorded, z1, z2, wrkstr);                          02017
   REF z1, z2, wrkstr; % the link %                                    02018
   IF mflag AND NOT recorded THEN                                      02019
      *wrkstr* _ "Message: " % no link %                                02020
   ELSE                                                                02021
      *wrkstr* _ "Location: ",  z1 z2;                                 02022
   RETURN(&wrkstr)                                                     02023
   END.
                                                                       02024
(gtollink)PROC(z1, z2);                                                02025
   REF z1, z2;                                                         02026
   RETURN(FIND SF(*lit*) [EOL] [EOL] ["Location: "] ['(] ^z1 _z1 ['')]
   ^z2);                                                               02027
   END.
                                                                       02028
(jnote)PROC(stid, wrkstr, iastr);                                      02029
   % avoid putting out "INFO-ONLY" indicator as a note.  If a string
   address is passed in iastr, then take "INFO-ONLY", "ACTION", or
   "Autor Copy" out of note and return it in iastr. %                  02030
   LOCAL classflag, classonlyflag;                                     02031
   LOCAL TEXT POINTER z1, z2;                                          02032
   REF wrkstr, iastr;                                                  02033
   IF FIND SF(stid) ["Note: "] ^z1 [";;"] ^z2 _z2 _z2 THEN             02034
      BEGIN                                                            02035
      *wrkstr* _ z1 z2;                                                02036
      FIND SF(*wrkstr*) ^z1;                                           02037
      classflag _ IF FIND (" [ INFO-ONLY ] ") ^z2 THEN 1               02038
         ELSE IF FIND (" [ ACTION ] " / "Autor Copy" ) ^z2 THEN 2
                                                                       02039
         ELSE 0;                                                       02040
      IF classflag THEN                                                02041
         BEGIN                                                         02042
         classonlyflag _ IF FIND ') THEN TRUE ELSE FALSE;              02043
         IF &iastr THEN *iastr* _ z1 z2;                               02044
         IF &iastr OR classflag=1 THEN                                 02045
            BEGIN                                                      02046
            IF classonlyflag THEN *wrkstr* _ NULL ELSE ST z1 z2 _
            NULL;                                                      02047
            END;                                                       02048
         END;                                                          02049
      IF wrkstr.L THEN                                                 02050
         *wrkstr* _ "*****Note: ", *wrkstr*, "*****", EOL;             02051
      END                                                              02052
   ELSE *wrkstr* _ NULL;                                               02053
   RETURN(&wrkstr);                                                    02054
   END.
                                                                       02055
% Network delivery citation format %                                   02056
   (nwkhed)PROCEDURE(adrstid,mflag,stid,l1,l2,recorded, wrkstr,
```

```
idfileno);                                                          02057
   LOCAL STRING iastr[30];                                          02058
   LOCAL TEXT POINTER z1;                                           02059
   REF 11, 12, wrkstr;                                              02060
   ON SIGNAL ELSE                                                   02061
      IF mflag THEN RETURN (FALSE);                                 02062
   FIND SF(stid) ^z1 ^jwp1;                                         02063
   *lit* _ "Date: ", *[nwkdatehcd($z1, &wrkstr)]*, EOL,             02064
      "Sender: ", *[nwkclerk($z1, &wrkstr, idfileno)]*, EOL,        02065
      "Subject: ", *[jtitle($z1, &wrkstr)]*, EOL,                   02066
      "From: ", *[nwkauthor($z1, &wrkstr)]*, EOL,                   02067
      "Message-id: ", *[nwkjrnloc($z1, &wrkstr, mflag, recorded, &11,
      &12)]*, EOL;                                                  02068
   jnote(adrstid, &wrkstr, $iastr);                                 02069
   IF iastr.L THEN *lit* _ *lit*, "Message-class: ", *iastr*, EOL;
                                                                    02070
   *lit* _ *lit*, EOL, %two crlf's terminate header% *wrkstr*;      02071
   IF [getjcomment(&wrkstr)].L <= 0 THEN *wrkstr* _ NULL            02072
   ELSE *wrkstr* _ "Comments: ", *wrkstr*;                          02073
   RETURN(TRUE);                                                    02074
   END.                                                             02075
(tzrec)RECORD                                                       02076
   tzdummy[18],                                                     02077
   tzfield[6]; %time zone field for idcnv reg 4%                    02078
(nwkdatehcd)PROCEDURE(z1, wrkstr);                                  02079
   LOCAL tzone, bbndate;                                            02080
   LOCAL TEXT POINTER z2, z3;                                       02081
   REF z1, wrkstr;                                                  02082
   bbndate.LH_1975; bbndate.RH_2; %March 75%                        02083
   CCPOS z1;                                                        02084
   IF FIND [";;;"] < 3CH $NP ^z3 [")"] > CH $NP ^z2 THEN            02085
      BEGIN                                                         02086
      *wrkstr* _ z2 z3, 0;                                          02087
      IF NOT FIND z3 < 3L THEN                                      02088
         BEGIN %no time zone, get it%                               02089
         IF FIND z1 > 2CH ("2" 4D / "1" 4D / 4D NLD) THEN tzone _ 5
         %East Coast% ELSE tzone _ 8 %westcoast% ;                  02090
         IF NOT SKIP !idtnc(&wrkstr+chbptr(0), 0) OR r4 < 0 THEN
         RETURN(&wrkstr);                                           02091
         IF r2 < bbndate THEN tzone _ 8; %before move to bbn%       02092
         r4.tzfield _ tzone;                                        02093
         IF SKIP !idcnv() THEN                                      02094
            BEGIN                                                   02095
            IF tops20flag THEN                                      02096
               BEGIN                                                02097
               !HRRZ r3,r2; %fraction of day * 1B6 in r3.RH%        02098
               !IMULI r3,250600B; %seconds in a day%                02099
               !ADDI r3,400000B; %rounding, seconds since midnight
               GMT in r3.LH%                                        02100
               !HLR r2,r3; %put back in r2.RH%                      02101
               END;                                                 02102
            *wrkstr* _ NULL;                                        02103
            datfrmt(r2, 12261B6, &wrkstr);                          02104
            IF FIND SE(*wrkstr*) $NP [SP] ^z2 THEN ST z2 z2 _ " At";
                                                                    02105
               %Put "At" between date and time%                     02106
```

```
            END;                                                    02107
         END;                                                       02108
      END                                                           02109
   ELSE *wrkstr* _ NULL;                                            02110
   RETURN(&wrkstr);                                                 02111
   END.                                                             02112
(nwkclerk)PROCEDURE(z1, wrkstr, idfilno);                           02113
   LOCAL host, subptr, backnloc;                                    02114
   LOCAL TEXT POINTER z2, z3, z4, z5;                               02115
   REF z1, wrkstr;                                                  02116
   subptr _ getsub(z1);                                             02117
   IF                                                               02118
      ( (FIND z1 > ([."Clerk:"])) OR (FIND SF(subptr) ["(Secondary
      Distribution Copy from "])  )                                 02119
      AND (FIND $NP LD ^z2 _z2 $LD ^z3)                             02120
      THEN                                                          02121
         BEGIN                                                      02122
         *wrkstr* _ z2 z3;                                          02123
         ckident(&wrkstr, $xlit, idfilno);                          02124
         getihost($xlit, 0, $z4, $z5);                              02125
         getinma($xlit, 0, $z2, $z3);                               02126
         *wrkstr* _ z2 z3, " At ", z4 z5;                           02127
         END                                                        02128
      ELSE                                                          02129
         BEGIN                                                      02130
         %don'T know what to do, use BACKGROUND At local host%     02131
         host _ IF lhostn=archost THEN $arcstr ELSE $utilstr;       02132
         backnloc _ IF lhostn=archost THEN $"BACKGROUND At " ELSE
         $"JDEMON At ";                                             02133
         *wrkstr* _ *[backnloc]*, *[host]*;                         02134
         END;                                                       02135
   RETURN(&wrkstr);                                                 02136
   END.                                                             02137
(nwkauthor)PROCEDURE(z1, wrkstr); %get author idents out of hcournal
tield by scanning to the date%                                      02138
   LOCAL TEXT POINTER z3;                                           02139
   REF z1, wrkstr;                                                  02140
   hjournal(&z1, &wrkstr);                                          02141
   IF FIND SF(*wrkstr*) [NP D] < CH $NP ^z3 THEN                    02142
      *wrkstr* _ SF(z3) z3                                          02143
   ELSE                                                             02144
      *wrkstr* _ NULL;                                              02145
   RETURN(&wrkstr);                                                 02146
   END.                                                             02147
(nwkjrnloc)PROCEDURE(z1, wrkstr, mflag, recorded, l1, l2);          02148
   LOCAL host;                                                      02149
   LOCAL TEXT POINTER z2, z3;                                       02150
   REF z1, wrkstr, l1, l2;                                          02151
   FIND z1 > 2CH ^z2 $D ^z3;                                        02152
   host _ IF lhostn=archost THEN $arcstr ELSE $utilstr;             02153
   *wrkstr* _ "[, *[host]*, "] JOURNAL ", z2 z3, " ";               02154
   IF NOT mflag AND FIND z1 > ["Hard Copy--Location: "] THEN        02155
      *wrkstr* _ *wrkstr*, "Hard Copy Only--";                      02156
   IF NOT mflag OR recorded THEN *wrkstr* _ *wrkstr*, ": ", l1 l2;
                                                                    02157
   RETURN(&wrkstr);                                                 02158
```

```
    END.                                                                02159
% Control support routines: check for ^S, full disc, load averages too
high %                                                                  02160
  (discfull)PROCEDURE(pgs);                                             02161
    %check load average and dismiss until < oljmlav; also check for ^S
    termination.  Called frequently enough at critical places to make
    this a valid method of termination.  Check if there is room on the
    disc to continue processing. %                                      02162
      lavchk();                                                         02163
    %check disc pages%                                                  02164
      r1_600000777777B;                                                 02165
      !JSYS gdskc; %214B%                                               02166
      IF pgs>r2 THEN RETURN(TRUE);                                      02167
    RETURN(FALSE);                                                      02168
    END.
                                                                        02169
  (lavchk)PROC; %dismiss until load average < oljmlav; also, check if
  control S has been typed.%                                            02170
    LOCAL sdstoptime;                                                   02171
    %convenient breakpt on ^S; this is called from a critical
    location%                                                           02172
      !SKIPE inpstp; stpchk();                                          02173
    %dismiss if load average is too high, terminate process if system
    going down soon%                                                    02174
      LOOP                                                              02175
        BEGIN                                                           02176
        IF SKIP !getab(lavtabad) AND r1 > oljmlav THEN BEGIN            02177
          %dismiss%                                                     02178
          r1 _ getsbn($"OJHLAV");                                       02179
          !JSYS setnm;                                                  02180
          r1 _ 100000; !JSYS disms; %100 seconds%                       02181
          r1 _ oljsbn; !JSYS setnm; %restore subsystem name%            02182
          REPEAT LOOP;                                                  02183
          END;                                                          02184
        IF (r1 _ gtadcall()) > nsdcktime THEN                           02185
          BEGIN %haven't checked for 5 minutes%                         02186
          gtadad5();                                                    02187
          nsdcktime _ r1; %gtad time 5 minutes from now%                02188
          gtadad5(); sdstoptime _ r1;                                   02189
          IF tops20flag THEN                                            02190
            BEGIN                                                       02191
            IF !sysgt(getsbn($"DWNTIM")) THEN  totentad();
            %convert r1 to tenex format%                                02192
            END                                                         02193
          ELSE                                                          02194
            BEGIN                                                       02195
            %tenex keeps downtime under systat%                         02196
            r1 _ lavtabad; r1.LH _ 27B;                                 02197
            IF NOT SKIP !getab() THEN r1 _ 0;                           02198
            END;                                                        02199
          IF r1 AND sdstoptime > r1 THEN SIGNAL (sigdsf,
          $"Terminated:  SYSTEM going down within 10 minutes");
                                                                        02200
          END;                                                          02201
        EXIT LOOP;                                                      02202
        END;                                                            02203
```

```
    RETURN;                                                          02204
    END.
                                                                     02205
(stpchk)PROC; %got ^S; terminate%                                    02206
     % change primary jfn to terminal and do DDT breakpoint %        02207
    LOCAL pofile;                                                    02208
    inpstp _ 0;                                                      02209
    !gpjfn(4B5);                                                     02210
    IF (pofile _ r2.RH) # 777777B THEN                               02211
       BEGIN                                                         02212
       r2.RH _ 777777B;                                              02213
       !spjfn();                                                     02214
       END;                                                          02215
    ddt(); %hits ddt breakpoint%                                     02216
    IF pofile # 777777B THEN %restore primary jfn%                   02217
       BEGIN                                                         02218
       !gpjfn(4B5);                                                  02219
       r2.RH _ pofile;                                               02220
       !spjfn();                                                     02221
       END;                                                          02222
    RETURN;                                                          02223
    END.
                                                                     02224
(gtadad5)PROC; %add 5 minutes to gtad time in r1%                    02225
    !ADDI r1,527654B;  %(1B6-(3600*24))+300%                         02226
    !TRNE r1,400000B;                                                02227
    !SUBI r1,527200B;  %didn't go through 2400 GMT%                  02228
    RETURN; END.
                                                                     02229
% Utility routines: driver file timing table setting and reading
procedures; time checking %                                          02230
   (jsettitab)PROCEDURE;                                             02231
      % Set Journal time table based on the TIMETABLE branch in JDRIVER
      file.  This table tells when the sleeping process is to be
      awakened and what is to be done upon awakening.  Also set load
      average cutoff if it has not beenset by hand. %                02232
      LOCAL jttstid, stid, i, mlav[2];                               02233
      LOCAL STRING lavstr[30];                                       02234
      LOCAL TEXTPOINTER z1, z2;                                      02235
      i _ 0;                                                         02236
      stid _ origin;                                                 02237
      ON SIGNAL ELSE                                                 02238
         snkerr($"Driver File open fail");                           02239
      stid.stfile _ open(0, jflname($"jdriver"));                    02240
      ON SIGNAL ELSE close(stid.stfile);                             02241
      IF (jttstid _ namelook(stid, $"timetable")) = endfil THEN      02242
         snkerr ($"No timetable in driver file");                    02243
      stid _ getsub(jttstid);                                        02244
      WHILE stid # jttstid DO                                        02245
         jnltime[i:=i+1] _ setjtw (stid:=getsuc(stid));              02246
      jnltime[i] _ 0; %terminator%                                   02247
      IF NOT liblod % load average set by hand % THEN                02248
         BEGIN                                                       02249
         IF (stid _ namelook(jttstid, $"mlav")) = endfil THEN snkerr
         ($"No mlav statement in driver file");                      02250
         FIND SF(stid) [']] [(':/)] ^z1 [('; <CH / ENDCHR)] ^z2;     02251
```

```
      *lavstr* _ z1 z2;                                                 02252
      nfloat($lavstr, $mlav, $mlav+1);                                  02253
      oljmlav _ mlav;                                                   02254
      END;                                                              02255
   close(jttstid.stfile);                                               02256
   RETURN;                                                              02257
   END.
                                                                        02258
(setjtw) PROCEDURE(stid); %process one statement to make one table
word for jnltime%                                                       02259
   LOCAL char, fchar, jnte, code;                                       02260
   LOCAL STRING tempsr[200];                                            02261
   jnte _ 0;                                                            02262
   *tempsr* _ SF(stid) SE(stid);                                        02263
   astruc($tempsr);                                                     02264
   FIND SF(*tempsr*);                                                   02265
   LOOP CASE (fchar _ READC) OF                                         02266
      ='P, ='N:                                                         02267
         BEGIN %Set the Period in hours into the INTERVAL (OJTIII)
         field; the Number of intervals into the NUMBER (OJTINN)
         field%                                                         02268
         r1 _ 0;                                                        02269
         FIND ['':];                                                    02270
         WHILE (char _ READC) # ''; DO                                  02271
            IF char IN ['0, '9] THEN r1 _ r1*10 + (char -'0);           02272
         CASE fchar OF                                                  02273
            ='P: jnte.ojtiii _ r1;                                      02274
            ='N: jnte.ojtinn _ r1;                                      02275
            ENDCASE;                                                    02276
         END;                                                           02277
      ='T:                                                              02278
         BEGIN % Set the time into the TIME (OJTITT) field%             02279
         r1 _ 0;                                                        02280
         FIND ['':];                                                    02281
         WHILE (char _ READC) # ''; DO                                  02282
            BEGIN                                                       02283
            IF char IN ['0, '9] THEN r1 _ r1*10 + (char -'0);           02284
            END;                                                        02285
         !IDIVI r1,100; %hours now in r1, minutes in r2%                02286
         !LSH r1,6;                                                     02287
         jnte.ojtitt _ r1 + ((r2 *64) + 30)/60; %time in 1/64ths of
         hour%                                                          02288
         END;                                                           02289
      ='D: % Set the tasks to be done at the appropriate wake up into
      the JFLG (OJTIJJJ) field%                                         02290
         BEGIN                                                          02291
         FIND ['':];                                                    02292
         code _ 0;                                                      02293
         LOOP CASE READC OF                                             02294
            ='0: code _ code .V 2B; % On-line distribution. %           02295
            ='S: code _ code .V 4B; % Slinker. %                        02296
            ='C: code _ code .V 10B; % Run detached, go to sleep,
            awake and Continue processing. %                            02297
            ='H: code _ code .V 20B; % Run Hard copy distribution. %
                                                                        02298
            =';: EXIT LOOP;                                             02299
```

```
            ENDCASE;                                                02300
         jnte.ojtijjj _ code;                                       02301
         END;                                                       02302
      =ENDCHR:  RETURN(jnte);                                       02303
      ENDCASE NULL;                                                 02304
   END.
                                                                    02305
(getojtime) PROCEDURE (ctime);                                      02306
   %given ctime = current time, find next time in this day to run
   on-line delivery.  returns 0 if ctime is after last delivery.
   times are in minutes from midnight%                              02307
   %Format of table jnltime entry: jjjiinnhhffB                     02308
      jjj: option flags: see joutil for meaning;  becomes libflg
                                                                    02309
      ii:  period (interval) length in hours                        02310
      nn: number of deliveries (at ii-hour intervals)               02311
      hh: hours field in time                                       02312
      ff: 64ths of hour field in time                               02313
   %                                                                02314
   LOCAL jcnt, jnte, jntes, val, btime, dcnt, inc;                  02315
   jcnt _ 0;                                                        02316
   val _ 777777B;                                                   02317
   WHILE (jnte _ jnltime [jcnt := jcnt + 1]) # 0 DO                 02318
      BEGIN                                                         02319
      btime _ ((jnte.ojtitt) * 60) / 64;                            02320
      dcnt _ (jnte.ojtinn) .A 77B;                                  02321
      IF NOT (inc _ jnte.ojtiii) THEN inc _ 1;                      02322
      inc _ inc * 60;                                               02323
      DO                                                            02324
         BEGIN                                                      02325
         IF btime > ctime AND btime < val THEN                      02326
            BEGIN                                                   02327
            val _ btime;                                            02328
            jntes _ jnte;                                           02329
            END;                                                    02330
         END                                                        02331
      UNTIL (dcnt _ dcnt - 1) <= 0 OR (btime := btime + inc) > ctime;
                                                                    02332
      END;                                                          02333
   IF val = 777777B THEN val _ jntes _ 0;                           02334
   RETURN(val % the time %, jntes.ojtijjj % The Journal utility flag
   %);                                                              02335
   END.
                                                                    02336
(timcheck)PROC;                                                     02337
   LOCAL cnt, cnti;                                                 02338
   cnt _ 0;                                                         02339
   LOOP                                                             02340
      BEGIN                                                         02341
      !JSYS gtad;                                                   02342
      IF r1 # -1 THEN RETURN;                                       02343
      IF (cnt:=cnt-1) <=0 THEN                                      02344
         BEGIN                                                      02345
         specttyout(oprtty, $"System does not have Date and Time
         set.");                                                    02346
         cnt _ 600; %complain about every 10 minutes%               02347
```

```
            END;                                                    02348
         FOR cnti _ 0 UP UNTIL >=400000B DO NULL;                   02349
            %disms gets illegal inst. trap if time not system time not
            set%                                                    02350
         END;                                                       02351
      END.
                                                                    02352
                                                                    02353
% Error handling.  Termination. %                                  02354
   (snkerr)PROC(astr);                                              02355
      %Accepts an error string in astr, blaps it to tty0, tty30, and
      primary output, an halts%                                     02356
      REF astr;                                                     02357
      %First set up subsystem name to snkerr%                       02358
         r1 _ getsbn($"SNKERR");                                    02359
         !JSYS setnm;                                               02360
      %Now blap error to tty 0 and 30%                              02361
         specttyout(oprtty, &astr);                                 02362
         specttyout(0, &astr);                                      02363
      %now type it to primary output%                               02364
         crlf();                                                    02365
         typeas(&astr);                                             02366
      %Now die%                                                     02367
         !JSYS haltf;                                               02368
      END.
                                                                    02369
% Network mail utility procedures. %                               02370
   (pokemailer)PROC(dirstr); %set bit in system file to tell mailer to
   get busy%                                                        02371
      LOCAL indx, blkadr, dirno, jfn;                               02372
      LOCAL STRING tempsr[70];                                      02373
      REF dirstr;                                                   02374
      jfn _ 0;                                                      02375
      ON SIGNAL ELSE                                                02376
         BEGIN                                                      02377
         IF jfn THEN sysclose(jfn, $tempsr);                        02378
         RETURN (FALSE);                                            02379
         END;                                                       02380
      %get directory number%                                        02381
         IF &dirstr AND dirstr.L > 0 THEN                           02382
            BEGIN                                                   02383
            *tempsr* _ *dirstr*;  %transdir adds a "0" to the string,
            so--%                                                   02384
            dirno _ transdir($tempsr);                              02385
            END                                                     02386
         ELSE                                                       02387
            BEGIN %use connected directory%                         02388
            !JSYS gjinf;                                            02389
            dirno _ r2;                                             02390
            END;                                                    02391
      %open mailer's flag file%                                     02392
         jfn _ sgtjfn( getgtjflg(read,0,oldvrsn),
         $"<SYSTEM>MAILER.FLAGS;", $tempsr );                       02393
         sysopen (jfn, rwthawed, random, $tempsr);                  02394
      %get a frozen core page%                                      02395
         frzblk ((indx _ lodrfb(0,-1)), 1);                         02396
```

```
      blkadr _ crpgad[indx];                                          02397
   % map in file page%                                                02398
      !HRLZ r1,jfn;                                                   02399
      r2 _ 4B11 .V (blkadr/1000B);                                    02400
      r3 _ 14B10;                                                     02401
      !JSYS pmap;                                                     02402
   %set the bit%                                                      02403
      r3 _ dirno.RH;                                                  02404
      !IDIVI r3,44B;                                                  02405
      r1 _ 4B11;                                                      02406
      r4 _ -r4;                                                       02407
      r3 _ r3 + blkadr;                                               02408
      !ROT r1,0(r4);                                                  02409
      !IORM r1,0(r3);                                                 02410
   %map out the page%                                                 02411
      r1 _ -1; %r2 is still set%                                      02412
      !JSYS pmap;                                                     02413
   %close mailer's file%                                              02414
      sysclose(jfn, $tempsr);                                         02415
   %thaw core page%                                                   02416
      frzblk(indx, -1);                                               02417
   RETURN(TRUE);                                                      02418
   END.
                                                                      02419
(openmail)PROC(userstr, hoststr, dirstr);                             02420
   LOCAL jfn;                                                         02421
   LOCAL STRING fnamstr[150], tempsr[80];                             02422
   REF userstr, hoststr, dirstr;                                      02423
   %make up name%                                                     02424
      IF &dirstr AND dirstr.L > 0 THEN *fnamstr* _ '<, *dirstr*, '>
      ELSE *fnamstr* _ NULL;                                          02425
      *fnamstr* _ *fnamstr*, 26B, "[--UNSENT-MAIL--", 26B, "].",
      *userstr*;                                                      02426
      IF hoststr AND hoststr.L > 0 THEN                               02427
         BEGIN                                                        02428
         astruc(&hoststr);                                            02429
         IF FIND SF(*hoststr*) ["SRI-ARC"] THEN *hoststr* _ *arcstr*;
                                                                      02430
         *fnamstr* _ *fnamstr*, 26B, '@, *hoststr*;                   02431
         END;                                                         02432
   %get stack and switch to it%                                       02433
      &ojsqsw_getsgw();                                               02434
      ojsqsw.swstid _ $fnamstr;                                       02435
      s _ ojsqsw.swsrsav := s;                                        02436
      m _ ojsqsw.swmrsav := m;                                        02437
      %calling parameters and locals can not be used while the stack
      is switched%                                                    02438
   ON SIGNAL ELSE                                                     02439
      BEGIN                                                           02440
      tda.dausgcod_ 0;                                                02441
      tda.davspec.vsusgf _ FALSE;                                     02442
      m _ ojsqsw.swmrsav;                                             02443
      s _ ojsqsw.swsrsav;                                             02444
      relsgw(&ojsqsw);                                                02445
      IF jfn _ sgtjfn(10000186, $fnamstr, $tempsr) THEN               02446
         BEGIN                                                        02447
```

```
         IF SKIP !sizef(jfn) AND r2=0 AND r3=0 THEN !delf(jfn);
         reljfn(jfn);
         END;
      END;
   tda.dausqcod_ $ojsqlink;
   tda.davspec.vsusqf _ TRUE;
   %the following call to opseqf causes the first port call through
   ojsqlink  which looks like a return to the caller of openmail%

      opseqf( ojsqsw.swstid, &tda, FALSE, FALSE, opmlfl);
   %the return to this point is affected by the final port call in
   closemail%
   tda.dausqcod_ 0;
   tda.davspec.vsusqf _ FALSE;
   ojsqsw.swstid _ FALSE;
   sport(&ojsqsw); %return to the close-out call of sport in
   closemail%
   END.

(mailstring)PROC(str, lev); %send a string to output sequential at
given level%
   ojsqsw.swstid _ asrref(str);
   ojsqsw.swclvl _ lev;
   sport (&ojsqsw); %port to ojsqlink%
   RETURN;
   END.

(mailBranch)PROC(stid, lev); %send a branch to output sequential at
given level%
   LOCAL substid;
   ojsqsw.swstid _ stid;
   ojsqsw.swclvl _ lev;
   sport (&ojsqsw); %port to ojsqlink%
   BUMP lev;
   substid _ getsub(stid);
   WHILE substid # stid DO mailbranch(substid := getsuc(substid),
   lev);
   RETURN;
   END.

(mailgroup) PROC(stid1, stid2, lev);
   DO mailbranch(stid1, lev) UNTIL (stid1:= getsuc(stid1)) = stid2;

   RETURN;
   END.
(ojsqlink)PROC(sw, calltype); %link between journal process and
output sequential%
   REF sw;
   CASE calltype OF
      =sqgnxt: NULL;
      ENDCASE RETURN;
   LOOP
      BEGIN
      ojsqsw.swstid _ TRUE; %return true%
      sport(&ojsqsw); %port to journal process%
      sw.swstid _ ojsqsw.swstid; %journal process' astring on its way
```

```
      to output sequential%                                              02491
      sw.swclvl _ ojsqsw.swclvl;                                         02492
      sport(&sw); %port to output sequential%                            02493
      END;                                                               02494
   END.
                                                                         02495
(closemail)PROC;                                                         02496
   ojsqsw.swstid _ endfil;                                               02497
   sport(&ojsqsw); %final port call into output sequential%              02498
   %is now all closed out, release seq work area%                        02499
   relsqw(&ojsqsw);                                                      02500
   RETURN;                                                               02501
   END.
                                                                         02502
(ftprcop)PROC(hostnam, hldir, hlpassw, hlaccnt, hext, dirnam, rtrext,
remvrdchar);                                                             02503
   LOCAL pos, jfn, errflg, ftstate, oftstate, checkq, filspec, ojfn;
                                                                         02504
   LOCAL STRING ifilnam[80], ofilnam[80], xfilnam[80], qfilnam[80],
   sqfilnam[80], nfilnam[80], lext[10], tempsr[80], jftperm[300],
   rtrspec[20], djunknam[25];                                            02505
   LOCAL TEXT POINTER p1, p2, p3, p4;                                    02506
   REF hostnam, hldir, hlpassw, hlaccnt, hext, dirnam, rtrext;           02507
   REF ftperm; %cell containg address of ftp err mess%                   02508
   checkq _ 0;                                                           02509
   jfn _ 0;                                                              02510
   errflg _ 0;                                                           02511
   oftstate_0;                                                           02512
   FIND SF(*hext*) ^p1 ([';] / [ENDCHR]) ^p2;                            02513
   *lext* _ p1 p2;                                                       02514
   (ftprstart):                                                          02515
   ojfn_0;                                                               02516
   ftstate_0;                                                            02517
   ON SIGNAL ELSE                                                        02518
      BEGIN                                                              02519
      IF a1=$ftpsig AND (FIND SF(*ftperm*) ("431 Accou") ) THEN GOTO
      ftprlog;                                                           02520
         %ignore "account not valid" error from ftplog%                  02521
      IF jfn THEN sysclose (jfn:=0, 0);                                  02522
      ON SIGNAL ELSE                                                     02523
         BEGIN                                                           02524
         ON SIGNAL ELSE snkerr(sysmsg);                                  02525
         ftpend();                                                       02526
         RETURN;                                                         02527
         END;                                                            02528
      *jftperm* _ "Journal FTP err: ", *[sysmsg]*;                        02529
      ftpcls();                                                          02530
      ON SIGNAL ELSE snkerr(sysmsg);                                     02531
      ftpend();                                                          02532
      IF NOT ftstate THEN RETURN;                                        02533
      IF ftstate = 2 AND NOT SKIP !delf(ojfn) THEN NULL;                 02534
      IF ftstate = 3 THEN                                                02535
         BEGIN                                                           02536
         %remote rename error, but file got accross ok%                  02537
         %must also test for local NLQE files%                           02538
         reljfn(jrnamf(ojfn, $ofilnam, $"NLQE;"));                       02539
```

```
      *jftperm* _ *jftperm*, CR, LF, "rename error on remote file
      ", *ifilnam*, CR, LF, "local file is ", *ofilnam*, CR, LF, "
      HELP!!";                                                    02540
      %type message to operator unless it's going to be done
      below%                                                      02541
         IF NOT errflg THEN                                       02542
            BEGIN                                                 02543
            specttyout(0, $jftperm); %to logging file%            02544
            specttyout(oprtty, $jftperm);                         02545
            END;                                                  02546
      END;                                                        02547
   reljfn(ojfn); %better check this is ok for zero jfn%           02548
   IF (errflg:=errflg+1) THEN %error abort, could test ftstate
   too%                                                           02549
      BEGIN                                                       02550
      IF ftstate=2 AND oftstate=2 THEN                            02551
         BEGIN                                                    02552
         *jftperm* _ *jftperm*, EOL, "happened twice on file: ",
         *ifilnam*;                                               02553
         IF NOT checkq THEN *sqfilnam* _ *ifilnam*; %was xfering
         file%                                                    02554
         END;                                                     02555
      %type message to operator and file%                         02556
         specttyout(0, $jftperm); %to logging file%               02557
         specttyout(oprtty, $jftperm);                            02558
      RETURN;                                                     02559
      END;                                                        02560
   oftstate_ftstate;                                              02561
   GOTO ftprstart;                                                02562
   END;                                                           02563
ttpbgn();                                                         02564
ftpopn (&hostnam);                                                02565
ttplog(&hldir, &hlpassw, &hlaccnt);                               02566
(ftprlog): %come here on "account not valid" signal%              02567
LOOP BEGIN                                                        02568
   ftstate_1;                                                     02569
   filspec _ IF checkq THEN $"nlq" ELSE $rtrext;                  02570
   *rtrspec* _ "*.", *[filspec]*, remvrdchar, '* ;                02571
   *djunknam* _ "djunk.txt", fvrdchar, '1 ;                       02572
   ftpdir($djunknam, $rtrspec);                                   02573
   IF NOT jfn _ lgetjfn (0, $djunknam, $txtext, gtjoisf, $lit)
   THEN                                                           02574
      callsig (ofilerr, $lit);                                    02575
   IF NOT sysopen (jfn, read, chrtyp, $lit) THEN                  02576
      BEGIN                                                       02577
      reljfn (jfn);                                               02578
      callsig (ofilerr, $lit);                                    02579
      END;                                                        02580
   WHILE isread (jfn, $ifilnam, LF) DO                            02581
      BEGIN                                                       02582
      ftstate_1;                                                  02583
      IF NOT (FIND SF(*ifilnam*) $LF ^p1 ( ('< ['>] ) / ) ^p4 ['.]
      ^p2 ['; / '.] ['';] ^p3 _p3) THEN EXIT;                     02584
      *ifilnam* _ p1 p3;                                          02585
      *qfilnam* _ p1 p2, "NLQ";                                   02586
      IF NOT checkq AND sqfilnam.L#0 AND *sqfilnam* = *ifilnam*
```

```
          THEN                                                          02587
             BEGIN                                                      02588
             ftpren ($ifilnam, $qfilnam);                               02589
             REPEAT LOOP;                                               02590
             END;                                                       02591
          *xfilnam* _ p1 p2, *lext*;                                    02592
          *ofilnam* _ *dirnam*, p4 p2, "NLO";                           02593
          IF NOT (ojfn _ sgtjfn(600101B6, $ofilnam, $tempsr)) THEN
          callsig(ofilerr, $tempsr);                                    02594
          jfnstr(ojfn, $ofilnam);                                       02595
          ftstate_2;                                                    02596
          ftprtr($ofilnam, $ifilnam, $"COMPRESSED");                    02597
          ftstate_3;                                                    02598
          ftpren ($ifilnam, $xfilnam);                                  02599
          ftstate_4;                                                    02600
          reljfn(jrnamf(ojfn, $ofilnam, &rtrext));                      02601
          errflg_ftstate_0;                                             02602
          END;                                                          02603
       IF jfn THEN sysclose(jfn:=0, 0);                                 02604
       IF (checkq:=checkq+1) THEN EXIT LOOP;                            02605
       END;                                                             02606
    ftpcls();                                                           02607
    ftpend();                                                           02608
    RETURN;                                                             02609
    END.                                                                02610
% Miscellaneous %                                                       02611
  (getnhcflag)PROC(submachine);                                         02612
    LOCAL flgs;                                                         02613
    flgs _ 0;                                                           02614
    IF submachine THEN flgs.delnet _ TRUE;                              02615
    IF lhostn=utilhost THEN flgs.delhc _ TRUE;                          02616
    RETURN(flgs);                                                       02617
    END.
                                                                        02618
  (setmastac)PROC; %set mastacprintflag; called by initialization
  routine.%                                                             02619
    mastacprintflag _ IF lhostn = utilhost THEN TRUE ELSE FALSE;        02620
    RETURN; END.
                                                                        02621
  (jrnamf)PROC(jfn,filenm,newext); % rename journal file with new
  extension. %                                                          02622
    %rename file with extension newext.  renames file with jfn "jfn"
    unless "jfn"=0, then uses name "filenm".  returns new string in
    "filenm".%                                                          02623
    LOCAL newjfn;                                                       02624
    LOCAL STRING tempsr [50];                                           02625
    LOCAL TEXT POINTER z1, z2, z3;                                      02626
    REF filenm, newext;                                                 02627
    IF jfn THEN jfnstr(jfn,&filenm) ELSE jfn _ sgtjfn(100101B6,
    &filenm,$tempsr);                                                   02628
    FIND SF(*filenm*) [".] ^z1 [ENDCHR] ^z2;                            02629
    FIND SE(*newext*) $NP ^z3 (("; / ".) ^z3 /  ) > ;                   02630
    ST z1 z2 _  SF(z3) z3;                                              02631
    IF NOT newjfn_sgtjfn(600101B6, &filenm,$tempsr) THEN
    snkerr($tempsr);                                                    02632
    r2_newjfn;                                                          02633
```

```
      r1_jfn;                                                          02634
      IF NOT SKIP !JSYS rnamf %35B% THEN BEGIN                         02635
         *tempsr* _ "RENAME err-file ", *filenm*;                      02636
         snkerr($tempsr);                                              02637
         END;                                                          02638
      RETURN (newjfn); END.
                                                                       02639
% ident list expansion %                                              02640
   (expdist)PROC(distid, p1); %expand idents pointed to by p1 into
   statement(s) down from distid%                                      02641
      LOCAL distexf, idfilno, going;                                   02642
      LOCAL STRING dstring[700];                                       02643
      REF p1;                                                          02644
      idfilno _ open(0, jflname($"identfile"));                        02645
      *dstring* _ NULL;                                                02646
      intids();                                                        02647
      DO BEGIN                                                         02648
         WHILE (going _ getids (&p1, $dstring, 1, idfilno)) AND
         dstring.L < 650 DO *dstring* _ *dstring*, ", ";               02649
         IF distexf THEN *dstring* _ *dstring*, ".PES;.HLT;";          02650
         distexf _ FALSE;                                              02651
         cis(distid, $dstring, $downstr);                              02652
         *dstring* _ NULL;                                             02653
         END UNTIL NOT going;                                          02654
      close (idfilno);                                                 02655
      RETURN;                                                          02656
      END.
                                                                       02657
                                                                       02658
   (iexpdist)PROC(delstr,idfnum);                                      02659
      LOCAL wrkstr;                                                    02660
      LOCAL TEXT POINTER ptr, z1, z2;                                  02661
      REF delstr, wrkstr;                                              02662
      &wrkstr _ getstring(1000, $dspblk);                              02663
      *delstr* _ *delstr*, ";";                                        02664
      makeptr(asrref(&delstr),$ptr);                                   02665
      *wrkstr* _ " ";                                                  02666
      intids(0);                                                       02667
      LOOP                                                             02668
         BEGIN                                                         02669
         *lit* _ NULL;                                                 02670
         IF NOT getids($ptr, $lit, 0, idfnum) THEN EXIT;               02671
         getiid($lit, 0, $z1, $z2);                                    02672
         *wrkstr* _ *wrkstr*, z1 z2, " ";                              02673
         END;                                                          02674
      *delstr* _ *wrkstr*;                                             02675
      freestring(&wrkstr, $dspblk);                                    02676
      RETURN;                                                          02677
      END.
                                                                       02678
                                                                       02679
FINISH jnldel                                                          02680
% Error handling Routines-- currently unused%                          02681
   (jerror)PROC(errval);                                               02682
      LOCAL count;                                                     02683
      jclosfl();                                                       02684
```

```
      IF errval < 0 THEN                                                02685
         BEGIN                                                          02686
         r1 _ chbptr(empty) + $errwrk;                                  02687
         r2 _ -errval .V 4B11;                                          02688
         r3 _ 0;                                                        02689
         !JSYS erstr;                                                   02690
         GOTO baderr;                                                   02691
         GOTO baderr;                                                   02692
         count _ 0;                                                     02693
         (baderr): dismes(2,$"System Error");                           02694
         END                                                            02695
      ELSE dismes(2,errval);                                            02696
      nlsrst();                                                         02697
      END.
                                                                        02698
%Procedures for initialising Journal Entry Mode-- currently unused%
                                                                        02699
   (jntrint) PROCEDURE;                                                 02700
      %This procedure checks the journal flags and sets the system name
      upon entrance to the journal.%                                    02701
      jdirn _ 0;                                                        02702
      IF jdebug THEN                                                    02703
         BEGIN                                                          02704
         dismes(1, $"Experimental Journal System--Do not use");         02705
         END                                                            02706
      ELSE                                                              02707
         IF jolock() THEN                                               02708
            BEGIN                                                       02709
            IF nlmode = fulldisplay THEN dismes(1, $"Deferred numbers
            only") ELSE typeas ($"(Deferred numbers only) ");           02710
            END;                                                        02711
      %set up subsystem name for statistics%                            02712
         IF NOT jnlsbn THEN jnlsbn _ <AUXCOD, getsbn>($jnlnam);         02713
         r1 _ jnlsbn;                                                   02714
         !JSYS setnm;                                                   02715
      RETURN;                                                           02716
      END.
                                                                        02717
   (jctlcint) PROCEDURE;                                                02718
      %This procedure performs the initialization required to service
      control C in the program.%                                        02719
      r1 _ 4B5;                                                         02720
      !JSYS rpcap;                                                      02721
      r3 _ r2;                                                          02722
      !JSYS epcap; %Permits process to assign ^C for pseudo interrupt%
                                                                        02723
      r1.LH _ 3;                                                        02724
      r1.RH _ 2;                                                        02725
      !JSYS ati;                                                        02726
      r1 _ 4B5;                                                         02727
      r2 _ 1B11;                                                        02728
      !JSYS aic; %Activate ^C interrupt channel = 2%                    02729
      conjdir(TRUE);                                                    02730
      jt0 _ jt1 _ jt2 _ jt3 _ jt4 _ jt5 _ jt6 _ jt7 _ jt8 _ jt9 _ jt10 _
      jt1 _ jt12 _ jt13 _ jt14 _ jt15 _ 0;                              02731
      RETURN;                                                           02732
```

```
    END.
                                                                    02733
%Routines for wrapping up JOurnal and returning to NLS-- currently
unused%                                                             02734
  (jreset)PROC;                                                     02735
    jclosfl();                                                      02736
    %toff 13%                                                       02737
    IF jtimfg THEN jtime();                                         02738
    nlsrst();                                                       02739
    END.
                                                                    02740
%Timing Procedures-- currently unused%                              02741
  (jtime)PROC;                                                      02742
    %Type out jthe jtimes of jthe run, and wait for a command accept
    before proceeding%                                              02743
    jtimout($"Open File", jt0);                                     02744
    jtimout($"open lock", jt1);                                     02745
    jtimout($"Open Work", jt2);                                     02746
    jtimout($"Set J Header", jt3);                                  02747
    jtimout($"Find Active", jt4);                                   02748
    jtimout($"enter J doc", jt5);                                   02749
    jtimout($"Update & delfn", jt6);                                02750
    jtimout($"Update JCAT", jt7);                                   02751
    jtimout($"Jcat Entry", jt8);                                    02752
    jtimout($"Dist Doc", jt9);                                      02753
    jtimout($"check Ident", jt11);                                  02754
    jtimout($"Set J Work", jt12);                                   02755
    jtimout($"Get Number", jt10);                                   02756
    jtimout($"Used Numbers", jt14);                                 02757
    jtimout($"Check Number", jt15);                                 02758
    jtimout($"Summed Total ", (jt0+jt1+jt2+jt3+jt4+jt5+jt6+jt7)
    +(jt8+jt9+jt10+jt11+jt12+jt14+jt15));                           02759
    jtimout($"Elapsed Total", jt13);                                02760
    crlf();                                                         02761
    typeas($"(Type CA)");                                           02762
    WHILE input() # CA DO NULL;                                     02763
    RETURN END.
                                                                    02764
  (jtimout)PROC(astr, value);                                       02765
    REF astr;                                                       02766
    LOCAL STRING st[20];                                            02767
    crlf();                                                         02768
    typeas(&astr);                                                  02769
    typeas($" = ");                                                 02770
    *st* _ STRING(value);                                           02771
    typeas($st);                                                    02772
    RETURN END.
                                                                    02773
%Katalog Klean-up-- currently unused%                               02774
  (getcfn) %fetch filename from catalog entry%
  PROCEDURE (entrystid, fldstr, ptr1, ptr2);                        02775
    %----------%                                                    02776
    LOCAL linkstid, retval;                                         02777
    LOCAL STRING dir [40], stmt [40], vspecs [40];                  02778
    LOCAL TEXT POINTER p1, p2;                                      02779
    REF fldstr;                                                     02780
```

```
    %----------%                                                    02781
    retval _ FALSE;                                                 02782
    IF NOT &fldstr THEN err ();                                     02783
    linkstid _ getsub (entrystid);                                  02784
    IF NOT (FIND SF(linkstid) ["Link"] ['('] ^p1 _p1 ['')'] ^p2) THEN
                                                                    02785
       getcend (entrystid, $p1, p2)                                 02786
    ELSE                                                            02787
       BEGIN                                                        02788
       lnkspec (1, $p1, $dir, &fldstr, $stmt, $vspecs);             02789
       *fldstr* _ *fldstr*, ".NLS";                                 02790
       retval _ TRUE;                                               02791
       END;                                                         02792
    stptset (0, ptr1, ptr2, $p1, $p2);                              02793
    RETURN (retval);                                                02794
    END.

                                                                    02795
                                                                    02796
(updtjo)PROC;                                                       02797
    %This procedure updates the pemanent copies of the Journal files
    JCAT an CNUMBERS from the temporary copies TCNUMBERS, TJCAT%    02798
    LOCAL jcstid, tjcstid, cstid, tcstid;                           02799
    sabtfg _ jcstid _ tjcstid _ cstid _ tcstid _ 0;                 02800
    jsavaccess _ access;                                            02801
    conjdir(TRUE);                                                  02802
    r1_getsbn($"JUPDCH");                                           02803
    !JSYS setnm;                                                    02804
    %**** STATE _ jcopst, spec; **** %                               02805
    IF sabtfg THEN supervisor()                                     02806
    ELSE BUMP sabtfg;                                               02807
    ON SIGNAL ELSE                                                  02808
       BEGIN                                                        02809
       undo(jcstid := 0);                                           02810
       undo(tjcstid := 0);                                          02811
       undo(cstid := 0);                                            02812
       undo(tcstid := 0);                                           02813
       access _ jsavaccess;                                         02814
       conjdir(FALSE);                                              02815
       r1_nlssbn;                                                   02816
       !JSYS setnm;                                                 02817
       END;                                                         02818
    enablaccess(0, jrnlaccess);                                     02819
    %First do the JCAT file%                                        02820
       jcstid _ openlock(0, jflname($"JCAT"));                      02821
       tjcstid _ openlock(0, jflname($"TJCAT"));                    02822
       updtpx(jcstid, tjcstid, $"Jcatalog", FALSE, FALSE);          02823
       updtpx(jcstid, tjcstid, $"catmods", TRUE, FALSE);            02824
       closeu((jcstid:=0).stfile);                                  02825
       closeu((tjcstid:=0).stfile);                                 02826
    %Now do cnumbers%                                               02827
       cstid _ openlock(0, jflname($"CNUMBERS"));                   02828
       tcstid _ openlock(0, jflname($"TCNUMBERS"));                 02829
       updtpx(cstid, tcstid, $"FREE", FALSE, FALSE);                02830
       updtpx(cstid, tcstid, $"USED", FALSE, FALSE);                02831
       updtpx(cstid, tcstid, $"INUSE", FALSE, FALSE);               02832
       updtpx(cstid, tcstid, $"PREASSIGNED", FALSE, TRUE);          02833
```

```
      closeu((cstid:=0).stfile);                                        02834
      closeu((tcstid := 0).stfile);                                     02835
   access _ jsavaccess; %restore access%                               02836
   conjdir(FALSE);                                                     02837
   rl_nlssbn;                                                          02838
   !JSYS setnm;                                                        02839
   supervisor();                                                       02840
   END.
                                                                       02841
(updtpx)PROC(stid1, stid2, namest, tailf, replfg);                     02842
   LOCAL dir, tstid1, tstid2;                                          02843
   LOCAL TEXT POINTER z1, z2, z3, z4;                                  02844
   REF namest;                                                         02845
   IF (tstid1 _ namelook(stid1, &namest)) = endfil THEN
   badfil(stid1.stfile);                                               02846
   IF (tstid2 _ namelook(stid2, &namest)) = endfil THEN
   badfil(stid2.stfile);                                               02847
   IF (stid2 _ getsub(tstid2)) # tstid2 THEN                           02848
      BEGIN %there is something to copy%                               02849
      IF NOT (tailf OR replfg) THEN                                    02850
         BEGIN                                                         02851
         stid1 _ tstid1;                                               02852
         dir _ levdown                                                 02853
         END                                                           02854
      ELSE                                                             02855
         IF ((stid1 _ getsub(tstid1)) = tstid1) THEN %nothing          02856
            under bracnh ... move to stid1 in levdown dir% dir _
            levdown                                                    02857
         ELSE                                                          02858
            BEGIN                                                      02859
            stid1 _ getail(stid1);                                     02860
            dir _ levsuc                                               02861
            END;                                                       02862
      IF replfg AND (stid1 # tstid1)  THEN                             02863
         crepgro(TRUE, gethed(stid1), getail(stid1), gethed(stid2),
         getail(stid2))                                                02864
      ELSE                                                             02865
         cmovgro(stid1, dir, gethed(stid2), getail(stid2), FALSE, 0);
                                                                       02866
      END                                                              02867
   ELSE                                                                02868
      IF (stid1 _ getsub(tstid1)) = tstid1 THEN                        02869
         BEGIN                                                         02870
         FIND SF(stid1) ^z1 SE(stid1) ^z2;                             02871
         FIND SF(stid2) ^z3 SE(stid2) ^z4;                             02872
         creptex($z1, $z2, $z3, $z4);                                  02873
         END;                                                          02874
   RETURN END.
                                                                       02875
(undo)PROC(stid);                                                      02876
   %un-lock and close file%                                            02877
   LOCAL fileno;                                                       02878
   IF (fileno _ stid.stfile) = 0 THEN RETURN;                          02879
   IF NOT writeout(fileno, $sar) OR NOT delpc(fileno, $sar)  THEN
                                                                       02880
      err($sar);                                                       02881
```

```
      close(fileno);                                                    02882
      RETURN END.

                                                                        02883
%more unused?%                                                          02884
   (hcdex)PROCEDURE;                                                    02885
      LOCAL distcount, collcount, niccount, char, collections,
      distribution, nic, pcap;                                          02886
      LOCAL TEXT POINTER z1;                                            02887
      LOCAL STRING passw[5], jnum[5], tempsr[5];                        02888
      %Hard copy distribution execute%                                  02889
      setmastac();                                                      02890
      %Parse start-up command%                                          02891
         %Password%                                                     02892
            *passw* _ NULL;                                             02893
            echoff();                                                   02894
            crlf();                                                     02895
            prompt($"Password:");                                       02896
            UNTIL passw.L = 3 DO                                        02897
               IF (char _ inpcuc()) = CD THEN                           02898
                  GOTO STATE                                            02899
               ELSE *passw* _ *passw*, char;                            02900
            IF *passw* # "JPD" THEN                                     02901
               BEGIN                                                    02902
               tqmarks();                                               02903
               GOTO STATE;                                              02904
               END;                                                     02905
         % Get operators ident. %                                       02906
            crlf();                                                     02907
            prompt($"Operator: ");                                      02908
            *opstr* _ NULL;                                             02909
            rdident($opstr, 0, 0);                                      02910
            %IF curchr # CA THEN getctc();%                             02911
      inithcd();                                                        02912
         %Control paameters%                                            02913
            distribution _ TRUE;                                        02914
            collections _ mastacprintflag;                              02915
            nic _ FALSE;                                                02916
         %if wheel, set priority and bring down load average cuttoff%
                                                                        02917
            IF (pcap _ enablw()) = -1 THEN BEGIN                        02918
               oljmlav _ 204400B6;                                      02919
               END                                                      02920
            ELSE BEGIN                                                  02921
               disablw (pcap := -1);                                    02922
               setpriority(202B);                                       02923
               oljmlav _ 202700B6;                                      02924
               END;                                                     02925
      %Type a warning message if this is the experimental system%       02926
         IF jdebug THEN                                                 02927
            BEGIN                                                       02928
            crlf();                                                     02929
            typeas($"Experimental Journal System");                     02930
            END;                                                        02931
      %Now parse control%                                               02932
         % **** STATE _ jdlstate, spec; **** %                         02933
         typech('$);                                                    02934
```

```
IF %tnumber($jnum)% FALSE > 0 THEN                                    02935
   BEGIN                                                              02936
   IF curchr # CA THEN error();                                       02937
   crlf();                                                            02938
   printnum($jnum, distribution, collections, nic); %
   number--output specific document%                                  02939
   END                                                                02940
ELSE CASE IF curchr IN ['a, 'z] THEN curchr-40B ELSE curchr OF
                                                                      02941
   ='A:                                                               02942
      BEGIN                                                           02943
      echo($"All");                                                   02944
      %getctc();%                                                     02945
      collections _ distribution _ nic _ TRUE;                        02946
      END;                                                            02947
   ='C: %Collection Copies%                                           02948
      BEGIN                                                           02949
      echo($"Collection Copies ");                                    02950
      collections _ answer();                                         02951
      END;                                                            02952
   ='D: %Distribution%                                                02953
      BEGIN                                                           02954
      echo($"Distribution Copies ");                                  02955
      distribution _ answer();                                        02956
      END;                                                            02957
   ='G: %Go%                                                          02958
      BEGIN                                                           02959
      echo($"Go");                                                    02960
      %getctc();%                                                     02961
      crlf();                                                         02962
      IF NOT (distribution OR collections OR nic) THEN err($"No
      Parameters Set");                                               02963
      rubabt _ 1;                                                     02964
      printall(distribution, collections, nic);                       02965
      END;                                                            02966
   ='H: %Hcdistfile collect%                                          02967
      BEGIN                                                           02968
      echo($"Hcdistfile garbage collect ");                           02969
      %getctc();%                                                     02970
      prntfg _ TRUE;                                                  02971
      jdquit();                                                       02972
      END;                                                            02973
   ='L: BEGIN                                                         02974
      mlavset();                                                      02975
      END;                                                            02976
   ='N: %Nic%                                                         02977
      BEGIN                                                           02978
      echo($"NIC Copies ");                                           02979
      nic _ answer();                                                 02980
      END;                                                            02981
   ='O: %Output Processor Version%                                    02982
      BEGIN                                                           02983
      echo($"Output Processor Version ");                             02984
      CASE inpcuc() OF                                                02985
         ='E, ='X:                                                    02986
            BEGIN                                                     02987
```

```
            echo($"Experimental");                                  02988
            %getctc();%                                             02989
            *oprocn* _ *xopname*;                                   02990
            END                                                     02991
         = 'N:                                                      02992
            BEGIN                                                   02993
            echo($"Normal");                                        02994
            %getctc();%                                             02995
            *oprocn* _ *opname*;                                    02996
            END                                                     02997
         ='?:                                                       02998
            help($"Experimental",                                   02999
               $"Normal",                                           03000
               0);                                                  03001
         ENDCASE tqmarks();                                         03002
   END;                                                             03003
='P:                                                                03004
   BEGIN                                                            03005
   echo($"Printer File Name ");                                     03006
   *lit* _ NULL;                                                    03007
   txtlit($lit);                                                    03008
   *ptstr* _ *lit*;                                                 03009
   END;                                                             03010
='Q:                                                                03011
   BEGIN                                                            03012
   echo($"Quit ");                                                  03013
   %getctc();%                                                      03014
   jdquit();                                                        03015
   reset();                                                         03016
   END;                                                             03017
='R: %reset parms%                                                  03018
   BEGIN                                                            03019
   echo($"Reset Parameters");                                       03020
   %getctc();%                                                      03021
   distribution _ collections _ nic _ FALSE;                        03022
   END;                                                             03023
='S:                                                                03024
   BEGIN                                                            03025
   echo($"Status");                                                 03026
   %getctc();%                                                      03027
   crlf();                                                          03028
   distcount _ hcstatus(:collcount, niccount);                      03029
   %type out distribution count%                                    03030
      crlf();                                                       03031
      typeas($"Distribution:  ");                                   03032
      *tempsr* _ STRING(distcount);                                 03033
      typeas($tempsr);                                              03034
   %Type out Collection count%                                      03035
      crlf();                                                       03036
      typeas($"Collection:  ");                                     03037
      *tempsr* _ STRING(collcount);                                 03038
      typeas($tempsr);                                              03039
   %Type out Nic count%                                             03040
      crlf();                                                       03041
      typeas($"Nic:  ");                                            03042
      *tempsr* _ STRING(niccount);                                  03043
```

```
            typeas($tempsr);                                              03044
         %type out total%                                                 03045
            crlf();                                                       03046
            typeas($"Total ");                                            03047
            *tempsr* _ STRING(distcount+collcount+niccount);              03048
            typeas($tempsr);                                              03049
         END;                                                             03050
      ="?:                                                                03051
         help($"Document Number",                                         03052
            $"All",                                                       03053
            $"Collection Copies ",                                        03054
            $"Distribuiton Copies ",                                      03055
            $"Go",                                                        03056
            $"NIC Copies ",                                               03057
            $"Output Processor Version ",                                 03058
            $"Printer File Name ",                                        03059
            $"Quit",                                                      03060
            $"Reset Parameters",                                          03061
            $"Status",                                                    03062
            0);                                                           03063
   ENDCASE tqmarks();                                                     03064
   GOTO STATE;                                                            03065
END.
                                                                          03066
```