```
< NLS, INPFBK.NLS;280, >, 14-OCT-74 10:33 DSM ;;;;
FILE inpfbk % L10 to <rel-nls>INPFBK % % (110,) (rel-nls,inpfbk.rel,) %
                                                                    02

%...declarations...%
                                                                    03
                                                                    04
   REF  rawchr;
   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
%...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
         qmoff();                                                   04748
      RETURN(curchr);                                               04757
      END.
                                                                    04758
   (lookc) PROCEDURE; %look at a character%                         028
      %look at next character in input buffer or tty%               029
      %return the character without advancing the buffer pointer%   030
      %-----------------%                                           031
      LOCAL char;                                                   032
```

```
   IF buffs = buffn % Is buffer empty? % THEN                          033
      BEGIN % must get another char from user %                        034
      buff[ buffn ] ← char ←                                           035
         IF nlmode = fulldisplay THEN dinptc() ELSE tinptc();          036
      IF (buffn ← buffn + 1) > buffsz THEN buffn ← 0;                  037
      END                                                              038
   ELSE                                                                039
      char ← buff[ buffs ];                                            040
   RETURN(char);                                                       041
   END.
                                                                       042
                                                                       043
(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
   IF nlcron AND nlstyp IN [0,2] THEN nlcrms(TRUE);                    068
   lpbin;                                                              069
   char ← r1;                                                          070
   IF nlcron AND nlstyp IN [0,2] THEN nlcrms(FALSE);                   071
   inptrf ← inpstp ← rubabt ← FALSE;                                   072
   RETURN(char);                                                       073
   END.
                                                                       074
(lpaltgetchar) PROCEDURE; % Lineprocessor GETCHAR routine %          06000
   %*****************************************************
   DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
   *****************************************************%           06001
   % this routine used for Line Processor terminals. It calls
   getchar and filters out special codes from LP for "system reset"
   and "printer string request" %                                   06002
   %-----------------%                                              06003
   LOCAL line, char, char2, lppcnt, lppfg;                          06004
   REF ltvsda;                                                      06686
   lppfg ← lppcnt ← 0;                                              06005
   LOOP                                                             06006
      BEGIN                                                         06637
      (lpalt2):                                                     06672
      IF altinp.L THEN                                              06639
         BEGIN                                                      06640
         char ← *altinp*[1];                                        06641
         *altinp* ← *altinp*[2 TO altinp.L];                        06642
         END                                                        06643
      ELSE                                                          06644
         BEGIN                                                      06645
         &rawchr ← $lpgetchar;                                      06646
         char ← getchar();                                          06647
         END;                                                       06648
      CASE char OF                                                  06007
         =176B: % special code sequence from LP %                   06008
            BEGIN                                                   06649
```

```
(lpaltl):                                                          06671
IF altinp.L THEN                                                   06651
   BEGIN                                                           06652
   char2 ← *altinp*[1];                                            06653
   *altinp* ← *altinp*[2 TO altinp.L];                             06654
   END                                                             06655
ELSE                                                               06656
   BEGIN                                                           06657
   &rawchr ← $lpgetchar;                                           06658
   char2 ← getchar();                                              06659
   END;                                                            06660
CASE char2 OF                                                      06009
   =177B: % user hit SYSTEM RESET %                                06010
      BEGIN                                                        06011
      % clear input buffer ??? %                                   06012
      %clear screen if sharing screens%                            06682
         IF ldspjfn THEN cscreen();                                06683
      IF (lptype .A 4M) = deltadata THEN                           06684
         cline(0, ltvsda.datop, lpxmax);                           06685
      !gjinf(); line ← r4;                                         06013
      lpcmode(); % put LP in coordinate mode %                     06017
      tracksend ← 0; % initialize lp printer %                     06254
      IF lppjfn THEN % printer was open %                          06018
         lppopen(); % open again %                                 06019
      ttywindow(ttyda); % setup TTY window %                       06020
      clrall(0, TRUE);  %assumes screen has been
      cleared%                                                     06217
      alldsp(); % repaint whole screen %                           06021
      dn($" ");                                                    06022
      dsubsys($ssysname);                                          06023
      dspvsp((vspsav:=0), vspsav[1], 3);                           06024
      char ← CD;                                                   06025
      EXIT LOOP;                                                   06026
      END;                                                         06027
   =40B: % LP requesting a string for printer %                    06028
      BEGIN                                                        06029
      IF altinp.L THEN                                             06661
         BEGIN                                                     06662
         char ← *altinp*[1];                                       06663
         *altinp* ← *altinp*[2 TO altinp.L];                       06664
         END                                                       06665
      ELSE                                                         06666
         BEGIN                                                     06667
         &rawchr ← $lpgetchar;                                     06668
         char ← getchar();                                         06669
         END;                                                      06670
      IF char = 176B THEN GOTO lpaltl;                             06030
      % should catch system restart in middle of stuff %
                                                                   06031
      lppcnt ← lppcnt + char - 40B;                                06032
      lppfg ← TRUE;                                                06033
      IF SKIP !sibe(dspjfn) THEN                                   06034
         BEGIN                                                     06035
         lppsr(lppcnt);                                            06036
         lppcnt ← 0;                                               06037
         lppfg ← 0;                                                06038
```

```
                    END;                                                    06039
                 END;                                                       06040
              =176B: % what the... %                                        06041
                 GOTO lpalt1; % try to do right thing %                     06042
              ENDCASE                                                       06043
                 err($"undefined LP code - lpgetchar");                     06044
           END;                                                             06650
        = 0: IF lppjfn THEN %printer request -- should have been a
        psi%                                                                06223
           BEGIN                                                            06227
           tracksend ← tracksend + 16;                                      06224
           GOTO lpalt2;                                                     06225
           END;                                                             06226
        ENDCASE EXIT LOOP;                                                  06045
     END;                                                                   06638
  IF lppfg THEN lppsr(lppcnt);                                              06046
  RETURN(char);                                                             06047
  END.                                                                      06048
(lpgetchar) PROCEDURE; % Lineprocessor GETCHAR routine %                    06580
  %*********************************************************
  DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
  *********************************************************%                06581
  % this routine used for Line Processor terminals. It calls
  getchar and filters out special codes from LP for "system reset"
  and "printer string request" %                                            06582
  %-------------------%                                                     06583
  LOCAL line, char, lppcnt, lppfg;                                          06584
  REF ltvsda;                                                               06681
  lppfg ← lppcnt ← 0;                                                       06585
  LOOP                                                                      06586
     CASE (char ←  getchar()) OF                                            06587
        =176B: % special code sequence from LP %                            06588
           CASE getchar() OF                                                06589
              =177B: % user hit SYSTEM RESET %                              06590
                 BEGIN                                                      06591
                 % clear input buffer %                                     06592
                    clrbuf(0);                                              07129
                 %clear screen if sharing screens%                          06676
                    IF ldspjfn THEN cscreen();                              06678
                 IF (lptype .A 4M) = deltadata THEN                         06679
                    cline(0, ltvsda.datop, lpxmax);                         06680
                 !gjinf(); line ← rh;                                       06593
                 IF tenex < 13200 THEN                                      06673
                    BEGIN                                                   06674
                    !sttyp(400000B+line,3); % set to TI to clear
                    buttons %                                               06594
                    !sttyp(400000B+line,13B); % back to LP %                06595
                    !sbcim( TRUE );  % do big char input %                  06596
                    intter(); %send interrogate command %                   06597
                    END;                                                    06675
                 lpcmode(); % put LP in coordinate mode %                   06598
                 tracksend ← 0; % initialize lp printer %                   06599
                 IF lppjfn THEN % printer was open %                        06600
                    lppopen(); % open again %                               06601
                 ttywindow(ttyda); % setup TTY window %                     06602
                 clrall(0, TRUE);  %assumes screen has been
```

```
                cleared%                                                06603
                alldsp(); % repaint whole screen %                      06604
                dn($" ");                                               06605
                dsubsys($ssysname);                                     06606
                dspvsp((vspsav:=0), vspsav[1], 3);                      06607
                char ← CD;                                              06608
                EXIT LOOP;                                              06609
                END;                                                    06610
             =40B: % LP requesting a string for printer %               06611
                BEGIN                                                   06612
                IF (char ← getchar()) = 176B THEN REPEAT CASE;
                                                                        06613
                % should catch system restart in middle of stuff %
                                                                        06614
                lppcnt ← lppcnt + char - 40B;                           06615
                lppfg ← TRUE;                                           06616
                IF SKIP !sibe(dspjfn) THEN                              06617
                   BEGIN                                                06618
                   lppsr(lppcnt);                                       06619
                   lppcnt ← 0;                                          06620
                   lppfg ← 0;                                           06621
                   END;                                                 06622
                END;                                                    06623
             =176B: % what the... %                                     06624
                REPEAT CASE; % try to do right thing %                  06625
             ENDCASE                                                    06626
                err($"undefined LP code - lpgetchar");                  06627
          = 0: IF lppjfn THEN %printer request -- should have been a
          psi%                                                          06628
             BEGIN                                                      06629
             tracksend ← tracksend + 16;                                06630
             REPEAT CASE;                                               06631
             END;                                                       06632
          ENDCASE EXIT LOOP;                                            06633
    IF lppfg THEN lppsr(lppcnt);                                        06634
    RETURN(char);                                                       06635
    END.                                                                06636
(nlcrms) PROCEDURE(ndofcom);  %Issue NLSCR JSYS%                        01268
    %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.%                                                         01269
    LOCAL curtim, extim;                                                01270
    r1 ← -5;                                                            01271
    !JSYS jobtm;                                                        01272
    extim ← r1; %result in millisecs%                                   01273
    !JSYS time;                                                         01274
    curtim ← r1; % result in milliseconds %                             01275
    IF ndofcom THEN                                                     01276
       BEGIN                                                            01277
       IF nlcrlst THEN %not first call to jsys%                         01278
          BEGIN                                                         01279
          r1 ← curtim - nlcrrt;                                         01280
          r2 ← extim - nlcret;                                          01281
          r3 ← curtim - (nlcrlst := curtim);                            01282
          r4 ← nlstyp;                                                  01283
```

```
            !JSYS nlscr;                                                    01284
            END                                                             01285
         ELSE nlcrlst ← curtim;                                             01286
         END                                                                01287
      ELSE                                                                  01288
         BEGIN                                                              01289
         nlcret ← extim;                                                    01290
         nlcrrt ← curtim;                                                   01291
         END;                                                               01292
      RETURN;                                                               01293
      END.
                                                                            01294
```

%...Display NLS Input Character Routines...%

```
                                                                            01175
   (dinptc) PROCEDURE; %DNLS single character input routine%                06061
      %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 is read.%                          06062
      %--------------------%                                                06063
      LOCAL char, da, tflag;                                                06064
      LOCAL STRING inmkr[50];                                               06065
      REF da;                                                               06066
      REF  rawchr;                                                          06067
      IF tenex < 13200 THEN                                                 06255
         BEGIN                                                              06256
         bugreg ← endfil;                                                   06068
         CASE char ← ( IF dinchr THEN dinchr := FALSE ELSE rawchr() )
         OF                                                                 06069
            < 200B: %normal character%                                      06070
               RETURN(trnsli[char]);                                        06071
            = mslwvws, = msupvws:  %viewspecs%                              06072
               BEGIN                                                        06073
               % for upper case viewspecs %                                 06074
                   tflag ← IF (char = mslwvws) THEN TRUE ELSE FALSE;
                                                                            06075
               %set up for viewspecs%                                       06076
                  IF NOT &da ← lda() THEN                                   06077
                     err($"No such Display Area");                          06078
                  dspvsp(sysvspec←da.davspec, sysvspec[1]←da.davspc2,
                  3);                                                       06079
                     %display the viewspecs to be affected%                 06080
                  ltl();  %large viewspecs%                                 06081
                  RESET savevspec;                                          06082
                  *vspstr* ← NULL;                                          06083
               LOOP %read the viewspec characters%                          06084
                  CASE char ← rawchr() OF                                   06085
                     =CD: % get out of viewspec mode %                      06086
                        BEGIN                                               06087
                        dendvw();                                           06088
                        RETURN(CD);                                         06089
                        END;                                                06090
                     < 200B: %normal character%                             06091
                        BEGIN                                               06092
                        char ← trnsli[char];                                06093
                        % for upper case viewspecs %                        06094
                            IF (NOT tflag) AND (char IN ['a,'z]) THEN
```

```
                                                                    06095
                    char ← char .A 137B;                            06096
                ON SIGNAL ELSE                                      06097
                    BEGIN                                           06098
                    ON SIGNAL ELSE;                                 06099
                    dendvw();                                       06100
                    END;                                            06101
                inpvsp(char);                                       06102
                ON SIGNAL ELSE;                                     06103
                IF inptrf THEN                                      06104
                    BEGIN                                           06105
                    dendvw();                                       06106
                    REPEAT CASE 2;                                  06107
                    END;                                            06108
                END;                                                06109
            = mslwvws, = msupvws:                                   06110
                BEGIN                                               06111
                tflag ← IF (char = mslwvws) THEN TRUE ELSE
                FALSE;                                              06112
                END;                                                06113
            = 200B: %end of viewspecs%                              06114
                BEGIN                                               06115
                dendvw();                                           06116
                EXIT LOOP;                                          06117
                END;                                                06118
            ENDCASE                                                 06119
                BEGIN                                               06120
                dendvw();                                           06121
                REPEAT CASE 2 (char);                               06122
                END;                                                06123
    REPEAT CASE; %get a real character for NLS%                     06124
    END;                                                            06125
= msmrkr:  %marker%                                                 06126
    BEGIN                                                           06127
    %set up for marker collection%                                  06128
        *inmkr* ← NULL; %clear inmkr string%                        06129
    LOOP                                                            06130
        CASE char ← rawchr() OF                                     06131
            =CD: % get out of marker collection %                   06132
                RETURN(CD);                                         06133
            < 200B: %normal character%                              06134
                *inmkr* ← *inmkr*, trnsli[char];                    06135
            = 200B: %end of marker%                                 06136
                EXIT LOOP;                                          06137
            ENDCASE                                                 06138
                BEGIN                                               06139
                dinchr ← char;                                      06140
                EXIT LOOP;                                          06141
                END;                                                06142
    IF inmkr.L NOT= empty THEN %get psid, character count
    from file marker table%                                         06143
        IF (bugreg ← lkmkr($inmkr, lcfile() : bugreg[1])) =
        endfil THEN                                                 06144
            err($"No such marker");                                 06145
    RETURN(CA);                                                     06146
    END;                                                            06147
```

```
        = 200B: REPEAT CASE;                                              06148
          %got undefined case shift input and aborted.  This is
          the end of the undefined case shift, so just get another
          character%                                                      06149
        ENDCASE                                                           06150
          BEGIN                                                           06151
          LOOP                                                            06152
            CASE char ← trnsli[rawchr()] OF                               06153
              =CD: RETURN(CD);                                            06154
              < 200B:                                                     06155
                BEGIN                                                     06156
                dismes(1000,$"undefined input");                          06157
                END;                                                      06158
              ENDCASE EXIT LOOP;                                          06159
          REPEAT CASE( char );                                            06160
          END;                                                            06161
    END                                                                   06257
ELSE                                                                      06258
    BEGIN                                                                 06259
    bugreg ← endfil;                                                      06261
    CASE char ← ( IF dinchr THEN dinchr := FALSE ELSE rawchr() )
    OF                                                                    06262
      # lpesc:                                                            06275
        CASE curmbt OF                                                    06276
          = 0: RETURN( trnsli[char] );                                    06277
          = 1: % right button down - markers %                            06278
            BEGIN                                                         06333
            msdbit ← TRUE;                                                06570
            %set up for marker collection%                                06334
              *inmkr* ← trnsli[char];                                     06335
            LOOP                                                          06341
              CASE char ← rawchr() OF                                     06342
                =CD: % get out of marker collection %                     06343
                  RETURN(CD);                                             06344
                # lpesc: %normal character%                               06345
                  *inmkr* ← *inmkr*, trnsli[char];                        06346
                ENDCASE                                                   06349
                  BEGIN                                                   06350
                  dinchr ← char;                                          06351
                  EXIT LOOP;                                              06352
                  END;                                                    06353
            IF inmkr.L NOT= empty THEN %get psid, character
            count from file marker table%                                 06354
              IF (bugreg ← lkmkr($inmkr, lcfile() :
              bugreg[1])) = endfil THEN                                   06355
                err($"No such marker");                                   06356
            RETURN(CA);                                                   06357
            END;                                                          06358
          = 2: % center button down - case shift 1 %                      06279
            BEGIN                                                         06285
            msdbit ← TRUE;                                                06571
            CASE char OF                                                  06266
              IN ['a, 'z]:  char ← char .A 137B;                          06287
              = ',: char ← '<;                                            06288
              = '.: char ← '>;                                            06289
              = ';: char ← ':;                                            06290
```

```
      = '?: char ← '\;                                          06291
      = SP: char ← TAB;                                         06292
      ENDCASE                                                   06293
         BEGIN                                                  06296
         dismes( 1000, $"undefined input");                     06297
         REPEAT CASE 3;                                         06298
         END;                                                   06299
   RETURN( trnsli[char] );                                      06294
   END;                                                         06295
= 3: % right & center buttons down %                            06280
   BEGIN                                                        06329
   msdbit ← TRUE;                                               06572
   dismes( 1000, $"undefined input");                           06330
   REPEAT CASE 2;                                               06331
   END;                                                         06332
= 4: % left button down - case shift 2 %                        06281
   BEGIN                                                        06310
   msdbit ← TRUE;                                               06573
   CASE char OF                                                 06311
      IN ['a, 'z]:  char ← cssh2[ char - 141B ];                06312
      = ',: char ← '[;                                          06313
      = '.: char ← '];                                          06314
      = ';: char ← '←;                                          06315
      = '?: char ← ALT;                                         06316
      = SP: char ← CR;                                          06317
      ENDCASE                                                   06318
         BEGIN                                                  06319
         dismes( 1000, $"undefined input");                     06320
         REPEAT CASE 3;                                         06321
         END;                                                   06322
   RETURN( trnsli[char] );                                      06323
   END;                                                         06324
= 5: % right & left buttons down %                              06282
   BEGIN                                                        06325
   msdbit ← TRUE;                                               06574
   dismes( 1000, $"undefined input");                           06326
   REPEAT CASE 2;                                               06327
   END;                                                         06328
= 6, % left & center buttons down - lower viewspecs %
                                                                06283
= 7: % all buttons down - upper case viewspecs %                06284
   BEGIN                                                        06359
   msdbit ← TRUE;                                               06575
   % for upper case viewspecs %                                06360
      tflag ← IF (curmbt = 6) THEN TRUE ELSE FALSE;
                                                                06361
   %set up for viewspecs%                                       06362
      IF NOT &da ← lda() THEN                                   06363
         err($"No such Display area");                           06364
      dspvsp(sysvspec←da.davspec,
      sysvspec[1]←da.davspc2, 3);                               06365
         %display the viewspecs to be affected%                 06366
      ltl();  %large viewspecs%                                 06367
      RESET savevspec;                                          06368
      *vspstr* ← NULL;                                          06369
   LOOP %read the viewspec characters%                          06370
```

```
                CASE char ← rawchr() OF                                 06371
                   =CD: % get out of viewspec mode %                    06372
                      BEGIN                                             06373
                      dendvw();                                         06374
                      RETURN(CD);                                       06375
                      END;                                              06376
                   # lpesc: %normal character%                          06377
                      BEGIN                                             06378
                      char ← trnsli[char];                              06379
                      % for upper case viewspecs %                      06380
                          IF (NOT tflag) AND (char IN ['a,'z])
                        THEN                                            06381
                           char ← char .A 137B;                         06382
                      ON SIGNAL ELSE                                    06383
                         BEGIN                                          06384
                         ON SIGNAL ELSE;                                06385
                         dendvw();                                      06386
                         END;                                           06387
                      inpvsp(char);                                     06388
                      ON SIGNAL ELSE;                                   06389
                      IF inptrf THEN                                    06390
                         BEGIN                                          06391
                         dendvw();                                      06392
                         REPEAT CASE 3;                                 06393
                         END;                                           06394
                      END;                                              06395
                   ENDCASE                                              06405
                      BEGIN                                             06406
                      IF (char ← dinbc()) = lpesc THEN                  06414
                         BEGIN                                          06415
                         dismes( 1000, $"Invalid Viewspec -
                         Ignored" );                                    06416
                         REPEAT CASE;                                   06417
                         END;                                           06418
                      CASE curmbt OF                                    06419
                         = 6, = 7:                                      06420
                            BEGIN                                       06424
                            tflag ←                                     06422
                               IF (curmbt = 6) THEN TRUE ELSE
                               FALSE;                                   06423
                            REPEAT CASE 2;                              06427
                            END;                                        06425
                         ENDCASE;                                       06421
                      dendvw();                                         06407
                      IF char THEN REPEAT CASE 3 (char)                 06408
                      ELSE REPEAT CASE 3;                               06429
                      END;                                              06409
             END;                                                       06411
          ENDCASE;                                                      06577
ENDCASE                                                                 06264
   IF char ← dinbc() THEN                                               06265
      BEGIN                                                             06578
      CASE char OF                                                      06431
         = lpesc: RETURN( trnsli[char] );                               06432
         ENDCASE REPEAT CASE (char);                                    06433
      END                                                               06579
```

```
            ELSE REPEAT CASE;                                          06413
      END;                                                             06260
   END.
                                                                       06162
(dinbc) PROCEDURE; % read a big character %                            06428
   LOCAL char, count;                                                  06434
   CASE char ← rawch°() OF                                             06435
      = lpesc: RETURN( char );                                         06436
      < 40B:                                                           06437
         BEGIN                                                         06438
         dismes( 1000, $"Illegal Input - Bad BCCNT");                  06439
         RETURN( FALSE );                                              06440
         END;                                                          06441
      ENDCASE count ← char - 40B;                                      06442
   WHILE count DO                                                      06444
      CASE char ← rawchr() OF                                          06448
         = fcor2:                                                      06449
            BEGIN                                                      06464
            IF count # 3 THEN                                          06463
               dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                                       06465
            WHILE (count ← count-1) DO rawchr();                       06466
            RETURN( FALSE );                                           06467
            END;                                                       06468
         = fcor4:                                                      06450
            BEGIN                                                      06469
            IF count # 5 THEN                                          06470
               dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                                       06471
            WHILE (count ← count-1) DO rawchr();                       06472
            RETURN( FALSE );                                           06473
            END;                                                       06474
         = acor2:                                                      06451
            IF count # 3 THEN                                          06476
               BEGIN                                                   06475
               dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                                       06477
               WHILE (count ← ccunt-1) DO rawchr();                    06478
               RETURN( FALSE );                                        06479
               END                                                     06480
            ELSE                                                       06481
               BEGIN                                                   06482
               gcords.xcord ← rawchr() - 40B;                          06484
               gcords.ycord ← rawchr() - 40B;                          06485
               RETURN( FALSE );                                        06486
               END;                                                    06483
         = acor4:                                                      06452
            IF count # 5 THEN                                          06487
               BEGIN                                                   06488
               dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                                       06489
               WHILE (count ← ccunt-1) DO rawchr();                    06490
               RETURN( FALSE );                                        06491
               END                                                     06492
            ELSE                                                       06493
               BEGIN                                                   06494
```

```
        gcords.xcord.xc1 ← rawchr() - 40B;                    06495
        gcords.xcord.xc2 ← rawchr() - 40B;                    06499
        gcords.ycord.yc1 ← rawchr() - 40B;                    06496
        gcords.ycord.yc2 ← rawchr() - 40B;                    06500
        RETURN( FALSE );                                      06497
        END;                                                  06498
= mbut2:                                                      06453
   IF count # 4 THEN                                          06501
      BEGIN                                                   06502
      dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                              06503
      WHILE (count ← count-1) DO rawchr();                    06504
      RETURN( FALSE );                                        06505
      END                                                     06506
   ELSE                                                       06507
      BEGIN                                                   06508
      ordmbt ← (curmbt ← rawchr() - 100B) .V ordmbt;          06513
      gcords.xcord ← rawchr() - 40B;                          06509
      gcords.ycord ← rawchr() - 40B;                          06510
      IF curmbt THEN RETURN( FALSE );                         06514
      char ← msetbl[ordmbt := 0];                             06515
      IF (msdbit := FALSE) THEN RETURN( FALSE )               06516
      ELSE RETURN( char );                                    06517
      END;                                                    06512
= mbut4:                                                      06454
   IF count # 6 THEN                                          06518
      BEGIN                                                   06519
      dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                              06520
      WHILE (count ← count-1) DO rawchr();                    06521
      RETURN( FALSE );                                        06522
      END                                                     06523
   ELSE                                                       06524
      BEGIN                                                   06525
      ordmbt ← (curmbt ← rawchr() - 100B) .V ordmbt;          06526
      gcords.xcord.xc1 ← rawchr() - 40B;                      06534
      gcords.xcord.xc2 ← rawchr() - 40B;                      06535
      gcords.ycord.yc1 ← rawchr() - 40B;                      06536
      gcords.ycord.yc2 ← rawchr() - 40B;                      06537
      IF curmbt THEN RETURN( FALSE );                         06529
      char ← msetbl[ordmbt := 0];                             06530
      IF (msdbit := FALSE) THEN RETURN( FALSE )               06531
      ELSE RETURN( char );                                    06532
      END;                                                    06533
= irep:                                                       06455
   IF count # 6 THEN                                          06538
      BEGIN                                                   06539
      dismes( 1000, $"Illegal Input - Bad BCCNT in BC");
                                                              06540
      WHILE (count ← count-1) DO rawchr();                    06541
      RETURN( FALSE );                                        06542
      END                                                     06543
   ELSE                                                       06544
      BEGIN                                                   06545
      lpxmax ← rawchr() - 40B;                                06547
      lpymax ← rawchr() - 40B;                                06548
```

```
                lptype ← rawchr() - 40B;                                    06549
                lpdlpad ← rawchr() - 40B;                                   06550
                lpbaudfactor ← rawchr() - 40B;                              06558
                RETURN( FALSE )                                             06557
                END;                                                        06555
            ENDCASE                                                         06456
               BEGIN                                                        06457
               dismes( 1000, $"Illegal Input - Bad Code in BC");            06458
               WHILE (count ← count-1) DO rawchr();                         06459
               RETURN( FALSE );                                             06460
               END;                                                         06461
      RETURN( FALSE );                                                      06447
      END.                                                                  06462

   (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%
                                                                        06687
  %read a literal from the keyboard, doing all of the control
  things, plus breaking on any control character identified by the
  string &chrlst%                                                       06688
  %types message if buffer overflows,                                   06689
  returns                                                               06690
     1 if normal termination (ca or c.),                                06691
     2 if a break character was input,                                  06692
     3 if a BC or BW was input and the string was already empty
                                                                        06693
  GLOBALS used for DNLS literal collection:                             06694
     litstore, litstring, tabstore, littabs, litline, litcolumn,
     spacecol                                                           06695
  %                                                                     06696
  %-----------------%                                                   06697
  LOCAL  char, nextchar, i, spclcharstr, breaktbl, display, nmfnd,
  exfnd, jfn, ind, bytptr, remchar, svcfo, sveso,
  locbreaktbl[128];                                                     06698
  LOCAL TEXT POINTER tp1, tp2, dp1, dp2,altd1,altd2,alte1,alte2;
                                                                        06699
  LOCAL STRING filstr[200], dirstr[200],extstr[10];                     06700
```

```
REF chrlst, astrng, spclcharstr, breaktbl;                          06701
litstring ← &astrng;                                                06702
*extstr* ← *nlsext*;                                                06703
FIND SF(*altdir*) ↑altdl;                                           06704
FIND SF(*altext*) ↑altel;                                           06705
display ← IF nlmode = fulldisplay THEN TRUE ELSE FALSE;             06706
%assumes monitor echoing for TNLS%                                  06707
   IF NOT display AND clpsw THEN echoff();                          06708
litstore ← chbptr(astrng.L) + &astrng;                              06709
   %for fast character appends and word-boundry line breaks%        06710
IF astrng.L AND NOT clpsw THEN                                      06711
   IF display THEN aplit(&astrng)                                   06712
   ELSE echo(&astrng); %echo knows about halfduplex%                06713
IF NOT clpsw AND NOT display AND buffs NOT= buffn THEN
typebuff();                                                         06714
   %typebuff knows about halfduplex%                                06715
IF (&chrlst AND chrlst.L) OR slink THEN  %must add user break
characters to system break charaters%                               06716
   BEGIN                                                            06717
   mvbfbf( $sysbreaktbl, $locbreaktbl, 128);                        06718
      %block transfer system table to local spcae%                  06719
   IF &chrlst AND chrlst.L THEN                                     06720
      BEGIN                                                         06721
      CCPOS SF(*chrlst*);                                           06722
      FOR i ← chrlst.L DOWN UNTIL <= 0 DO                           06723
         BEGIN                                                      06724
         char ← READC;                                              06725
         locbreaktbl[char] ← $userbreak;                            06726
         END;                                                       06727
      END;                                                          06728
   IF slink THEN                                                    06729
      BEGIN                                                         06730
      locbreaktbl['<↑F>] ← $rdlctf;                                 06731
      locbreaktbl['<ESC>] ← $rdlesc;                                06732
      IF NOT display THEN                                           06733
         BEGIN                                                      06734
         svcfo ← trnslo['<↑F>] := nullch;                           06735
         svesc ← trnslo['<ESC>] := nullch;                          06736
         END;                                                       06737
      END;                                                          06738
   &breaktbl ← $locbreaktbl;                                        06739
   END                                                              06740
ELSE &breaktbl ← $sysbreaktbl;                                      06741
LOOP  %process a character at a time%                               06742
   BEGIN                                                            06743
   IF nldevice = devlproc AND NOT tracking AND SKIP !sibe(100B)
   THEN                                                             06744
      track();                                                      06745
   char ← input();                                                  06746
   GOTO [breaktbl[char]];                                           06747
   (rdladdchar): %normal character%                                 06748
      IF astrng.L >= astrng.M THEN                                  06749
         BEGIN                                                      06750
         dismes(2, $"literal too long -- CDOT simulated");          06751
         curchr ← char ← C.;                                        06752
         GOTO rdlcacdot;                                            06753
```

```
      END;                                                          06754
    %*astrng* ← *astrng*, char;%                                    06755
      ↑litstore ← char;                                             06756
      BUMP astrng.L;                                                06757
    IF display AND NOT clpsw THEN litchr(char); %echo char in
    literal area%                                                   06758
    REPEAT LOOP;                                                    06759
(rdlbc):      %backspace character%                                 06760
    IF astrng.L = empty THEN RETURN(3, char)                        06761
    ELSE                                                            06762
      BEGIN                                                         06763
      char ← .litstore; %last char input%                           06764
      IF NOT display AND NOT clpsw THEN %echo what is being
      backed over%                                                  06765
        BEGIN                                                       06766
        CASE char OF                                                06767
          IN [SP, 'z]: %normal character%                           06768
            ipbout(char);                                           06769
          ENDCASE %Special non-printing character%                  06770
            BEGIN                                                   06771
            &spclcharstr ← npstrad(char);                           06772
            typeas(&spclcharstr);                                   06773
            END;                                                    06774
        END;                                                        06775
      BUMP DOWN astrng.L;                                           06776
      %back up litstore%                                            06777
        r1 ← litstore;                                              06778
        r2 ← litstore-1;                                            06779
        UNTIL r2 = r1 DO                                            06780
          BEGIN                                                     06781
          r3 ← r2;                                                  06782
          !IBP r2;                                                  06783
          END;                                                      06784
        litstore ← r3;                                              06785
      IF display AND NOT clpsw TREN  %back up literal area%
                                                                    06786
        litbc(char, .litstore);                                     06787
      END;                                                          06788
    REPEAT LOOP;                                                    06789
(rdlbw):      %backspace word%                                      06790
    IF astrng.L = empty THEN RETURN(3, char)                        06791
    ELSE                                                            06792
      BEGIN                                                         06793
      CCPOS SE(*astrng*);                                           06794
      i ← astrng.L;                                                 06795
      nextchar ← READC;                                             06796
      CASE char ← nextchar OF                                       06797
        = ENDCHR: NULL;                                             06798
        = NP;                                                       06799
          BEGIN %space past non letters/digits%                     06800
          BUMP DOWN i;                                              06801
          IF display THEN                                           06802
            BEGIN                                                   06803
            IF NOT clpsw THEN litbc(char, nextchar ←
            READC);                                                 06804
            REPEAT CASE;                                            06805
```

```
                END;                                                    06806
              nextchar ← READC;                                         06807
              REPEAT CASE;                                              06808
              END;                                                      06809
           ENDCASE;                                                     06810
        CASE char ← nextchar OF                                         06811
           = ENDCHR: NULL;                                              06812
           = NLD:                                                       06813
              IF char = PT THEN                                         06814
                 BEGIN %space past non letters/digits%                  06815
                 BUMP DOWN i;                                           06816
                 IF display THEN                                        06817
                    BEGIN                                               06818
                    IF NOT clpsw THEN litbc(char, nextchar ←
                    READC);                                             06819
                    REPEAT CASE;                                        06820
                    END;                                                06821
                 nextchar ← READC;                                      06822
                 REPEAT CASE;                                           06823
              END;                                                      06824
           ENDCASE;                                                     06825
        CASE char ← nextchar OF                                         06826
           = LD:                                                        06827
              BEGIN %space past letters/digits%                         06828
              BUMP DOWN i;                                              06829
              IF display AND NOT clpsw THEN litbc(char, ENDCHR
              );                                                        06830
              nextchar ← READC;                                         06831
              REPEAT CASE;                                              06832
              END;                                                      06833
           ENDCASE;                                                     06834
        astrng.L ← i;                                                   06835
        litstore ← chbptr(astrng.L) + &astrng;  %for fast
        character appends%                                              06836
        END;                                                            06837
     REPEAT LOOP;                                                       06838
(userbreak):  %user provided break character%                           06839
     IF NOT display THEN                                                06840
        BEGIN                                                           06841
        IF clpsw THEN echon();                                          06842
        IF slink THEN                                                   06843
           BEGIN                                                        06844
           trnslo['<↑F>] ← svcfo;                                       06845
           trnslo['<ESC>] ← sveso;                                      06846
           END;                                                         06847
        END;                                                            06848
     RETURN(2, char);                                                   06849
(rdlcacdot):  %command accept or center dot%                            06850
     IF NOT display THEN                                                06851
        BEGIN                                                           06852
        IF clpsw THEN echon();                                          06853
        IF slink THEN                                                   06854
           BEGIN                                                        06855
           trnslo['<↑F>] ← svcfo;                                       06856
           trnslo['<↑F>] ← svcfo;                                       06857
           END;                                                         06858
```

```
      END;                                                          06859
   RETURN(1, char);                                                 06860
(rdlbs):      %backspace statement%                                06861
   *astrng* ← NULL;                                                 06862
   litstore ← chbnty + &astrng;  %for fast character appends%       06863
   IF display THEN                                                  06864
      setlit()                                                      06865
   ELSE                                                             06866
      BEGIN                                                         06867
      !pbout(EOL);                                                  06868
      !pbout(EOL);                                                  06869
      END;                                                          06870
   REPEAT LOOP;                                                     06871
(rdlretype):  %retype literal%                                      06872
   IF NOT clpsw THEN                                                06873
      IF display THEN                                               06874
         BEGIN                                                      06875
         rstlit();                                                  06876
         aplit(&astrng);                                            06877
         END                                                        06878
      ELSE                                                          06879
         BEGIN                                                      06880
         !pbout(EOL);                                               06881
         !pbout(EOL);                                               06882
         typeas(&astrng);                                           06883
         END;                                                       06884
   REPEAT LOOP;                                                     06885
(rdlle):      %literal escape%                                      06886
   CASE (char ← rawchr()) OF                                        06887
      = '<↑F>, = '<ESC>:                                            06888
         IF slink THEN                                              06889
            BEGIN                                                   06890
            GOTO [breaktbl[char]];                                  06891
            END                                                     06892
         ELSE REPEAT CASE( ENDCHR );                                06893
      ENDCASE                                                       06894
         BEGIN                                                      06895
         IF NOT clpsw AND NOT display AND char NOT IN[ SP, 'z]
         THEN                                                       06896
            BEGIN                                                   06897
            &spclcharstr ← npstrad(char);                           06898
            typeas(&spclcharstr);                                   06899
            END;                                                    06900
         GOTO rdladdchar;                                           06901
         END;                                                       06902
(rdlesc): NULL;       % altmode - handle like control-F %          06903
(rdlctf):     % control-F %                                         06904
   remchar ← IF char = '<↑F> THEN TRUE ELSE FALSE;                  06905
   *filstr* ← NULL;                                                 06906
   FOR and ← 1 UP UNTIL > filstr.M  DO                              06907
      *filstr* ← *filstr*, 0;                                       06908
   FIND SE(*astrng*) ↑tp2                                           06909
      [ ', > CH $(SP/TAB) / ENDCHR > $(SP/TAB) ] ↑tp1;              06910
   *filstr* ← tp1 tp2, '<ESC>;                                      06911
   *dirstr* ← NULL;                                                 06912
```

```
IF FIND tpl < $(SP/TAB) ', $(SP/TAB) ↑dp2 (-ENDCHR AND -',)
                                                        06913
   [ ', > CH $(SP/TAB) / ENDCHR > $(SP/TAB) ] ↑dp1      06914
     THEN *dirstr* ← + dp1 dp2, 0;                      06915
(rdlcfi):                                               06916
IF cdlnk AND (jfn ←                                     06917
   lgetjfn( IF dirstr.L THEN $dirstr ELSE 0,            06918
      $filstr, $extstr, gtjidl+ (IF nwlnk THEN gtjoof ELSE
      gtjcif), $lit) ) THEN                             06919
         BEGIN                                          06920
         jfntostr( jfn, $dirstr, 001110B6+1);           06921
         IF NOT SKIP !rljfn( jfn ) THEN NULL;           06922
         *filstr*[filstr.L] ← 0;  BUMP DOWN filstr.L;   06923
         nmfnd ← exfnd ← FALSE;                         06924
         FOR ind ← 1 UP UNTIL > filstr.L DO             06925
            CASE *filstr*[ind] OF                       06926
               = '.: nmfnd ← TRUE;                      06927
               = ';: exfnd ← TRUE;                      06928
               ENDCASE;                                 06929
         IF NOT nmfnd THEN                              06930
            FOR ind UP UNTIL > dirstr.L DO              06931
               IF (char ← *dirstr*[ind]) # '. THEN      06932
                  BEGIN                                 06933
                  IF NOT rdlutl( &astrng, $char, display)
                                                        06934
                     THEN GOTO rdlcacdot;               06935
                  END                                   06936
               ELSE                                     06937
                  BEGIN                                 06938
                  BUMP ind;                             06939
                  IF NOT rdlutl( &astrng, $char, display)
                                                        06940
                     THEN GOTO rdlcacdot;               06941
                  IF remchar THEN REPEAT LOOP 2 ELSE EXIT
                  LOOP;                                 06942
                  END;                                  06943
         IF NOT exfnd THEN                              06944
            FOR ind UP UNTIL > dirstr.L DO              06945
               IF (char ← *dirstr*[ind]) # '; THEN      06946
                  BEGIN                                 06947
                  IF NOT rdlutl( &astrng, $char, display)
                                                        06948
                     THEN GOTO rdlcacdot;               06949
                  END                                   06950
               ELSE                                     06951
                  BEGIN                                 06952
                  BUMP ind;                             06953
                  IF NOT rdlutl( &astrng, $char, display)
                                                        06954
                     THEN GOTO rdlcacdot;               06955
                  IF remchar THEN REPEAT LOOP 2 ELSE EXIT
                  LOOP;                                 06956
                  END;                                  06957
            FOR ind UP UNTIL > dirstr.M DO              06958
               IF (char ← *dirstr*[ind]) # ENDCHR THEN  06959
                  BEGIN                                 06960
```

```
                IF NOT rdlutl( &astrng, $char, display)  06961
                   THEN GOTO rdlcacdot;                 06962
                END                                     06963
             ELSE                                       06964
                BEGIN                                   06965
                char ← ',;                              06966
                IF NOT rdlutl( &astrng, $char, display) 06967
                   THEN GOTO rdlcacdot;                 06968
                EXIT LOOP;                              06969
                END;                                    06970
          END                                           06971
ELSE                                                    06972
   BEGIN                                                06973
   % try the alternate directory and alternate extensions%
                                                        06974
   %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.%      06975
   IF altdfl  AND FIND altdl > [',] ↑altd2 THEN         06976
      BEGIN                                             06977
      *dirstr* ←  + altdl altd2 ;                       06978
      *dirstr*[dirstr.L] ← 0;                           06979
      BUMP DOWN dirstr.L;                               06980
      IF FIND altel > [',] ↑alte2  THEN                 06981
         BEGIN                                          06982
         *extstr* ← + altel alte2;                      06983
         *extstr*[extstr.L] ← 0;                        06984
         BUMP DOWN extstr.L;                            06985
         altel[1] ← alte2[1];                           06986
         GOTO rdlcfl;                                   06987
         END                                            06988
      ELSE                                              06989
         BEGIN                                          06990
         altdl[1] ← altd2[1] ;                          06991
         FIND SF(*altext*) ↑altel;                      06992
         GOTO rdlcfl;                                   06993
         END;                                           06994
      END;                                              06995
   % if failed due to bad directory, try sans directory %
                                                        06996
      IF cdlnk AND (rl = $gjfx17) AND (dirstr.L := 0) THEN
                                                        06997
         GOTO rdlcfl;                                   06998
   % construct string for STDIR %                       06999
      *filstr*[filstr.L] ← 0;                           07000
      filstr.L ← (ind ← filstr.L) - 1;                  07001
   % contruct byte pointer to directory name for STDIR %
                                                        07002
      bytptr ← chbmty + $filstr;                        07003
   % get directory number given directory name %        07004
      !stdir( 4^B10, bytptr, 0);                        07005
         GOTO notdir;       % ambiguous and no match %  07006
         GOTO notdir;       % like normal input %       07007
```

```
                GOTO gotdir;        % got a good number now %    07008
            (notdir):                                           07009
               IF NOT clpsw AND NOT display THEN                07010
                  BEGIN                                         07011
                  &spclcharstr ← npstrad(char);                 07012
                  typeas(&spclcharstr);                         07013
                  END;                                          07014
               GOTO rdladdchar;                                 07015
            (gotdir=):                                          07016
               filstr.L ← filstr.M;                             07017
               FOR ind UP UNTIL > filstr.M DO                   07018
                  IF (char ← *filstr*[ind]) THEN                07019
                     BEGIN                                      07020
                     IF NOT rdlutl( &astrng, $char, display)    07021
                        THEN GOTO rdlcacdot;                    07022
                     END                                        07023
                  ELSE                                          07024
                     IF cdlnk THEN                              07025
                        BEGIN                                   07026
                        char ← ',;                              07027
                        IF NOT rdlutl( &astrng, $char, display) 07028
                           THEN GOTO rdlcacdot;                 07029
                        EXIT LOOP;                              07030
                        END                                     07031
                     ELSE EXIT LOOP;                            07032
            END;                                                07033
         REPEAT LOOP;                                           07034
      END;                                                      07035
   END.                                                         07036
                                                                07037
(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
%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%                                                        03634
      %-------------------%                                    03635
      LOCAL STRING send[10];                                    05892
      CASE nldevice OF                                          03636
         = deviproc:                                            03637
            CASE char OF                                        03638
```

```
        = TAB:                                                  04223
          BEGIN                                                 04562
          IF tracking THEN                                      04563
             position(litda.daleft + litcolumn[litline]-1,
             litda.datop + litline);                            04564
          !bout(dspjfn, lptab);                                 04565
          IF ldspjfn THEN %linked to another devlproc%          04566
             !bout(ldspjfn, lptab);                             04567
          END;                                                  04568
        = EOL:                                                  04558
          BEGIN                                                 04559
          position(litda.daleft, litda.datop+litline+1);
                                                                04560
          END;                                                  04561
        = BC:                                                   04224
          BEGIN                                                 04225
          IF litcolumn[litline] <= 1 THEN %must back up to
          previous line and delete last char%                   04239
             position(litda.daleft + (IF litline >= 1 THEN
             litcolumn[litline-1] ELSE 0), litda.datop +
             MAX(0, litline-1))                                 04240
           ELSE IF tracking THEN                                04242
             position(litda.daleft + litcolumn[litline]-1,
             litda.datop + litline);                            04243
          !bout(dspjfn, BC);                                    04226
          !bout(dspjfn, SP);                                    04227
          !bout(dspjfn, BC);                                    04234
          IF ldspjfn THEN %linked to another devlproc%          04228
             BEGIN                                              04229
             !bout(ldspjfn, BC);                                04235
             !bout(ldspjfn, SP);                                04236
             !bout(ldspjfn, BC);                                04237
             END;                                               04232
          END;                                                  04233
        ENDCASE                                                 03648
          BEGIN                                                 03649
          IF tracking THEN                                      04244
             position(litda.daleft + litcolumn[litline]-1,
             litda.datop + litline);                            04245
          !bout(dspjfn, char);                                  03650
          IF ldspjfn THEN %linked to another devlproc%          03651
             !bout(ldspjfn, char);                              03652
          END;                                                  03653
    = imlac0, = imlac1:                                         05769
      BEGIN                                                     05770
      *send* ← begmsg, 4+remfudge, apsda, litdahandle.daidr1 +
      remfudge, litdahandle.daidr2 + remfudge;                  05771
      IF char = EOL                                             05772
         THEN BEGIN                                             05773
            *send* ← *send*, CR, LF;                            05774
            *send*[2] ← 5+remfudge;                             05775
            END                                                 05776
         ELSE *send* ← *send*, char;                            05777
      !sout(dspjfn, chbmty + $send, -send.L);                   05778
      IF linkcnsl THEN                                          05779
         !sout(ldspjfn, chbmty + $send, -send.L);               05780
```

```
            END;                                                    03781
         ENDCASE                                                    03654
            BEGIN                                                   03655
            r1 ← 0;                                                 03656
            r1.sdaid ← litdahandle;                                 03657
            IF NOT SKIP !strda(r1, 707B8 .V $char, 7B8 .V $char, 0)
            THEN                                                    03658
               IF r1 = syserr THEN NULL                             03659
               ELSE                                                 03660
                  dismes(2,$"NLS Display Error, litpbout");         03661
            IF linkcnsl THEN                                        03662
               BEGIN                                                03663
               r1 ← 0;                                              03664
               r1.sdaid ← litda.dalhandle;                          03665
               IF NOT SKIP !strda(r1 %assumes other registers still
               ok%) THEN                                            03666
                  IF r1 = syserr THEN NULL                          03667
                  ELSE                                              03668
                     dismes(2,$" Link -  NLS Display Error,
                     litpbout");                                    03669
               END;                                                 03670
            END;                                                    03671
      RETURN;                                                       03672
      END.
                                                                    03673
(litchr) PROCEDURE (char); %append char to literal display area%
                                                                    03082
      %--------------------%                                       03083
      LOCAL                                                         03209
         spclcharstr, %address of special character string%         03396
            %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(frdtab(&tda, litcolumn/litline/)-1, litrmarg);
                                                                    03090
            rtabstore ← 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%                                                        03231
      LOCAL i, spclcharstr;                                          03382
      REF spclcharstr;                                               03383
      litpbout(BC);                                                  03304
      CASE char OF                                                   03258
         IN [SP,'z]:  %normal character%                             03259
            IF (litcolumn[litline] ← litcolumn[litline] -1) < 1 THEN
                                                                     03260
```

```
        %go to previous litline%                                    03261
        IF litline = 0 THEN litcolumn[litline] ← 1                  03262
        ELSE                                                        03263
           BEGIN                                                    03264
           litline ← litline-1;                                     03265
           CASE prevchar OF                                         03310
              = EOL, = LF, = CR: NULL;                              03311
              = ENDCHR: NULL;                                       03348
              ENDCASE                                               03312
                 BEGIN                                              03313
                 litpbout(BC); litpbout(prevchar);                  03266
                    %necessary because of the way sequential
                    display areas treats EOL, LF, and CR%
                                                                    03267
                 END;                                               03314
           END;                                                     03268
= TAB:                                                              03269
    BEGIN                                                           03270
    i ← .tabstore;                                                  03271
    FOR i DOWN UNTIL <= 0 DO litpbout(BC);                          03273
    i ← .tabstore + 1;                                              03379
    %back up tabstore%                                              03274
       r1 ← tabstore;                                               03275
       r2 ← tabstore-1;                                             03276
       UNTIL r2 = r1 DO                                             03277
          BEGIN                                                     03278
          r3 ← r2;                                                  03279
          !IBP r2;                                                  03280
          END;                                                      03281
       tabstore ← r3;                                               03282
    IF (litcolumn[litline] ← litcolumn[litline] - i) < 1
    THEN                                                            03272
       %go to previous litline%                                     03315
       IF litline = 0 THEN litcolumn[litline] ← 1                   03316
       ELSE                                                         03317
          BEGIN                                                     03318
          litline ← litline-1;                                      03319
          CASE prevchar OF                                          03320
             = EOL, = LF, = CR: NULL;                               03321
             = ENDCHR: NULL;                                        03349
             ENDCASE                                                03322
                BEGIN                                               03323
                litpbout(BC); litpbout(prevchar);                   03324
                   %necessary because of the way sequential
                   display areas treats EOL, LF, and CR%
                                                                    03325
                END;                                                03326
          END;                                                      03327
    END;                                                            03283
= EOL, = LF:                                                        03284
    BEGIN                                                           03285
    IF litline = 0 THEN litcolumn[litline] ← 1                      03286
    ELSE                                                            03330
       BEGIN                                                        03331
       litline ← litline-1;                                         03332
       CASE prevchar OF                                             03333
```

```
                    = EOL, = LF, = CR: NULL;                                03334
                    = ENDCHR: NULL;                                         03350
                    ENDCASE                                                 03335
                       BEGIN                                                03336
                       litpbout(prevchar);                                  03337
                          %necessary because of the way sequential
                          display areas treats EOL, LF, and CR%             03338
                       END;                                                 03339
                 END;                                                       03340
              END;                                                          03290
           = CR:                                                            03291
              BEGIN                                                         03292
              CASE prevchar OF                                              03341
                 = EOL, = LF, = CR: NULL;                                   03342
                 = ENDCHR: NULL;                                            03351
                 ENDCASE                                                    03343
                    BEGIN                                                   03344
                    litpbout(BC); litpbout(prevchar);                       03345
                       %necessary because of the way sequential
                       display areas treats EOL, LF, and CR%                03346
                    END;                                                    03347
              END;                                                          03295
           ENDCASE  %non-printing, special character%                       03296
              BEGIN                                                         03297
              &spclcharstr ← npstrad(char);                                 03298
                 %get address of string which represents this
                 character%                                                 03380
              FOR i ← spclcharstr.L-1 DOWN UNTIL <=0 DO                     03299
                 litpbout(BC);                                              03300
              IF (litcolumn[litline] ← litcolumn[litline] -
              spclcharstr.L) < 1 THEN                                       03352
                 %go to previous litline%                                   03353
                 IF litline = 0 THEN litcolumn[litline] ← 1                 03354
                 ELSE                                                       03355
                    BEGIN                                                   03356
                    litline ← litline-1;                                    03357
                    CASE prevchar OF                                        03358
                       = EOL, = LF, = CR: NULL;                             03359
                       = ENDCHR: NULL;                                      03360
                       ENDCASE                                              03361
                          BEGIN                                             03362
                          litpbout(BC); litpbout(prevchar);                 03363
                             %necessary because of the way sequential
                             display areas treats EOL, LF, and CR%
                                                                            03364
                          END;                                              03365
                    END;                                                    03366
              END;                                                          03301
        RETURN;                                                             03303
        END.                                                                03305
                                                                            03306
%...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);                                                 04766
    LOCAL tchar, spclcharstr;                                        04770
    REF spclcharstr;                                                 04771
    %-----------------%                                              04772
    tchar ← trnslo(char);                                            04773
    CASE tchar OF                                                    04774
       = nullch,                                                     04767
       = 'Y-100B : NULL;                                             04768
       = TAB,                                                        04769
       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 feedbk. 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, $outstr + chcnty);                             05981
       GOTO xsndbad;                                               05982
       GOTO xsndbad;                                               05983
       dirno ← r1.RH;                                              05984
       retflg ← FALSE;                                             05985
       IF messtring.L > 99 THEN messtring.L  < 99;                  05986
       *outstr* ← EOL, *messtring*;                                05987
    % look for match(es) on this user - currently logged in and
    attached %                                                     05988
       FOR i ← 1 UP UNTIL > 77B DO                                 05989
          BEGIN                                                    05990
          r1.LH ← i;  % job # %                                    05991
          r1.RH ← jobtbl; %global%                                 05992
          IF NOT SKIP !getab(r1) THEN RETURN(retflg); %probably end
          of table%                                                05993
          IF dirno = r1.RH THEN retflg ← csndtms($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
      rl.LH ← jobno;                                                04815
      rl.RH ← ttytbl;  %global%                                     04816
      % get tty number %                                            04817
         IF NOT SKIP !getab(rl) THEN RETURN(FALSE);                 04818
         IF (ttyno ← rl) = -1 THEN RETURN(FALSE); %detached%        04819
         !sout(4B5 .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
         rl ← 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
      rl ← 18M; % controlling tty %                                 0760
      !JSYS rfmod; % read mode %                                    0761
      r2 ← r2 .A 777777001777B;                                     0762
      r2 ← r2 .V 17Bh; %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
      rl ← 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
      rl ← 18M; % controlling tty %                                 0770
      !JSYS rfmod; % read mode %                                    0771
      r2 ← r2 .A 777777001777B;                                     0772
```

```
        r2 ← r2 .V 16483;                                                   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 17283; %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(cm.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.rtxl ← 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%                              03704
   %This procedure removes all of the marks (currently circles)
   that provide feedback of bug selects on the screen.%                03705
   %-----------------%                                                 03706
   LOCAL lsid, daid, ldaid, save, bmempty, char, x, y;                 03707
   LOCAL STRING send[20];                                              05782
   IF NOT bmcnt THEN RETURN;  %no bug marks on screen%                 03708
   save ← bugmarks;                                                    03709
   RESET bugmarks;                                                     03710
   bmempty ← bugmarks := save;                                         03711
   CASE nldevice OF                                                    03712
      = deviproc:                                                      03713
         BEGIN                                                         03714
         UNTIL bugmarks = bmempty DO                                   03715
            BEGIN                                                      03716
            lppopmark();                                               03720
            POP bugmarks TO y;                                         06177
            POP bugmarks TO x;                                         06178
            POP bugmarks TO char;                                      06175
            *send* ← char;                                             06176
            IF char NOT= SP THEN                                       06179
               lpdspstr(x, y, $send, FALSE);                           06180
            END;                                                       03721
         END;                                                          03722
      = imlac0, = imlac1:                                              05783
         BEGIN                                                         05784
         UNTIL bugmarks = bmempty DO                                   05785
            BEGIN                                                      05786
            POP bugmarks TO lsid;                                      05787
            POP bugmarks TO ldaid;                                     05788
            POP bugmarks TO daid;                                      05789
            *send* ← begmsg, 5+remfudge, remssda,
            daid.daidr1+remfudge, daid.daidr2+remfudge, lsid,
            TRUE;                                                      05790
            !sout(dspjfn, chbmty + $send, -send.L);                    05791
            IF ldaid THEN                                              05792
               !sout(ldspjfn, chbmty + $send, -send.L);                05793
            END;                                                       05794
         END;                                                          05795
      ENDCASE                                                          03723
         BEGIN                                                         03724
         UNTIL bugmarks = bmempty DO                                   03725
            BEGIN                                                      03726
            POP bugmarks TO lsid;                                      03727
            POP bugmarks TO ldaid;                                     03728
            POP bugmarks TO daid;                                      03729
            r1.sstrid ← lsid;                                          03730
            r1.sdaid ← daid;                                           03731
            r2 ← r3 ← r4 ← 0;                                          03732
            IF NOT SKIP !JSYS strda THEN                               03733
               NULL;                                                   03734
            IF ldaid THEN                                              03735
               BEGIN                                                   03736
               r1.sstrid ← lsid;                                       03737
               r1.sdaid ← ldaid;                                       03738
               r2 ← r3 ← r4 ← 0;                                       03739
```

```
                IF NOT SKIP !JSYS strda THEN                          03740
                   NULL;                                             03741
                END;                                                 03742
             END;                                                    03743
          END;                                                       03744
    bmcnt ← 0;                                                       03745
    RETURN;                                                          03746
    END.
                                                                     03747
(delbm) PROCEDURE; %turn the most recent bug mark off%               03748
    %if the most recent bug  was via a marker, or if there are no
    bug marks on the screen, this routine just returns%              03749
    %------------------%                                            03750
    LOCAL lsid, daid, ldaid, save, bmempty, char, x, y;              03751
    LOCAL STRING send[20];                                           05796
    IF NOT bmcnt THEN RETURN;  %no bug marks on screen%              03753
    save ← bugmarks;                                                 03754
    RESET bugmarks;                                                  03755
    bmempty ← bugmarks := save;                                      03756
    CASE nldevice OF                                                 03757
       = devlproc:                                                   03758
          IF bugmarks NOT= bmempty THEN                              03759
             BEGIN                                                   03760
             lppopmark();                                            03764
             POP bugmarks TO y;                                      06170
             POP bugmarks TO x;                                      06171
             POP bugmarks TO char;                                   06169
             *send* ← char;                                          06174
             IF char NOT= SP THEN                                    06172
                lpdspstr(x, y, $send, FALSE);                        06173
             END;                                                    03765
       = imlac0, = imlac1:                                           05797
          IF bugmarks NOT= bmempty THEN                              05798
             BEGIN                                                   05799
             POP bugmarks TO lsid;                                   05800
             POP bugmarks TO ldaid;                                  05801
             POP bugmarks TO daid;                                   05802
             *send* ← begmsg, 5+remfudge, remssda,
             daid.daidr1+remfudge, daid.daidr2+remfudge, lsid, 1;
                                                                     05803
             !sout(dspjfn, chbmty + $send, -send.L);                 05804
             IF ldaid THEN                                           05805
                !sout(ldspjfn, chbmty + $send, -send.L);             05806
             END;                                                    05807
       ENDCASE                                                       03766
          IF bugmarks NOT= bmempty THEN                              03767
             BEGIN                                                   03768
             POP bugmarks TO lsid;                                   03769
             POP bugmarks TO ldaid;                                  03770
             POP bugmarks TO daid;                                   03771
             r1.sstrid ← lsid;                                       03772
             r1.sdaid ← daid;                                        03773
             r2 ← r3 ← r4 ← 0;                                       03774
             IF NOT SKIP !JSYS strda THEN                            03775
                NULL;                                                03776
             IF ldaid THEN                                           03777
```

```
                    BEGIN                                                   03778
                    r1.sstrid ← lsid;                                       03779
                    r1.sdaid ← ldaid;                                       03780
                    r2 ← r3 ← r4 ← 0;                                       03781
                    IF NOT SKIP !JSYS strda THEN                            03782
                       NULL;                                                03783
                    END;                                                    03784
                 END;                                                       03785
      BUMP DOWN bmcnt;                                                      03786
      RETURN;                                                               03787
      END.
                                                                            03788
%...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.rtxl, ls.rtx2] THEN                          02415
              DIV (xcrdnt - ls.rtxl)/ls.rthinc, chrcnt, diff             02416
           ELSE                                                          02417
              IF (diff ← ls.rtxl - xcrdnt) > 0 THEN                      02418
                 chrcnt ← 0                                              02419
              ELSE                                                       02420
                 BEGIN                                                   02421
                 diff ← xcrdnt -ls.rtx2;                                 02422
                 chrcnt ← (ls.rtx2 - ls.rtxl)/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.rtxl+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%                                                                 03789
      %The string that tracks the mouse position is set by this
      routine %                                                             03790
      %----------------%                                                    03791
      LOCAL bp;                                                             03792
      LOCAL STRING send[30];                                                05808
      REF string;                                                           03793
      CASE nldevice OF                                                      03794
        = devlproc: NULL;                                                   03795
        = imlac0, = imlac1:                                                 05809
          BEGIN                                                             05810
          *send* ← begmsg, 5+string.L+remfudge, remscsr,
          string.L+1, csize, hinc, font, *string*;                          05811
          !sout(dspjfn, chbmty + $send, -send.L);                           05812
          IF linkcnsl THEN                                                  05813
            !sout(ldspjfn, chbmty + $send, -send.L);                        05814
          END;                                                              05815
        ENDCASE                                                             03796
          BEGIN                                                             03797
          bp ← chbptr(empty) + &string;                                     03798
          r2 ← chbptr(string.L) + &string;                                  03799
          r1 ← bp;                                                          03800
          r3.scfont ← font;                                                 03801
          r3.schinc ← hinc;                                                 03802
          r3.sccsiz ← csize;                                                03803
          IF NOT SKIP !JSYS scsr THEN                                       03804
            err($"SCSR JSYS error, csrstr");                                03805
          IF r4 ← linkcnsl THEN                                             03806
            BEGIN                                                           03807
            r1 ← bp;                                                        03808
            IF NOT SKIP !JSYS nscsr THEN                                    03809
              err($"Link NSCSR JSYS error, csrstr");                        03810
            END;                                                            03811
          END;                                                              03812
      RETURN;                                                               03813
      END.
                                                                            03814
%...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.rtxl DO             04247
            *vsstr2* ← *vsstr2*, SP;                              04248
         lpdspstr(llsrt.rtxl+vspcda.daleft,
         llsrt.rty+vspcda.datop, $vsstr2, NOT ltsmall1);          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
(lt1) 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...%                                                    01119
(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($tmpstr);                                         06212
      RETURN;                                                  06214
      END;                                                     06215
   &llsrt ← namda.dalsrt;                                      03951
   olength ← dnstr.L;                                          03952
   *dnstr* ← NULL;                                             03953
   chrcnt ← 0;                                                 03954
   UNTIL (chrcnt ← chrcnt + 1) > astrng.L OR chrcnt > dnstr.M DO
                                                               03965
      BEGIN                                                    03956
      CASE char ← *astrng*(chrcnt) OF                          03957
         =CA, =C., =LF, =CD, =EC, =BW, =$ascalt, IN (1B, 32B), IN
         (3AB, 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
      (mrkl) PROCEDURE; %put CFL up arrow in first pos%        03986
```

```
   %----------------%                                                  03987
   LOCAL ls;                                                          03988
   REF ls;                                                            03989
   &ls ← cflda.dalsrt + lsrtl;                                        03990
   ls.rtxl ← 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.rtxl;                       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.rtxl < newpos THEN             04003
         BEGIN %must erase begining of the old one%                   04210
         length ← (newpos - llsrt.rtxl)/llsrt.rthinc;                 04212
         cline (cflda.daleft+llsrt.rtxl,
         cflda.datop+llsrt.rty, length);                              04004
         END;                                                         04211
      llsrt.rtxl ← newpos;                                            04005
      llsrt.rtx2 ← MAX(llsrt.rtx2, llsrt.rtxl +
      (cflarw.L-1)*(llsrt.rthinc));                                   04006
      wrtstr (&cflda, &llsrt);                                        04622
      llsrt.rtx2 ← llsrt.rtxl + (cflarw.L-1)*(llsrt.rthinc);
                                                                      04213
      END;                                                            04629
   RETURN;                                                            04009
   END.
                                                                      04010
(mlfl) 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, savexl;                                    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.rtxl, max + ls.rtxl - 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
       savexl ← ls.rtxl;                                            05893
       ls.rtxl ← ls.rtxl + (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.rtxl ← savexl;                                            05894
       ls.rtx2 ← MAX( ls.rtxl, cflstr.L + ls.rtxl - 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%           04419
       LOCAL end, da, ls, endls, savelitline;                       04420
       LOCAL STRING send[20];                                       05816
       REF da, ls;                                                  04421
       IF nlmode # fulldisplay THEN RETURN;                         05425
       IF litreset THEN RETURN;                                     04422
       %clear the literal feedback area %                           04423
          savelitline ← litline;                                    04424
          clearlit();                                               04425
       CASE nldevice OF                                             04426
          = devlproc:                                               04427
             BEGIN                                                  04428
             %take care of tty windows!!%                           04429
                IF littyflag := FALSE THEN %restore normal tty
                window%                                             04430
                   BEGIN                                            04431
                   lpttywindow(msgda.datop, msgda.dabottom);        04432
                   nlmode ← fulldisplay;                            04433
                   litapflag ← TRUE;                                04434
                   savelitline ← litda.dabottom-litda.datop-1;      04435
                   END;                                             04436
             %restore suppressed file text display if LITAPFLAG set
             -- otherwise let DAUPDATE or DAFRMT do it (with an
             appropriate initialization of the dalsupp fileds!)%    04437
                IF (litapflag := FALSE)  AND savelitline > 0 THEN
                %must restore old text%                             04438
                   BEGIN                                            04439
                   end ← &dupyarea + dal*dacnt;                     04440
```

```
            FOR &da ← $dpyarea UP dal UNTIL >= end DO          04441
               IF  da.daexis AND NOT da.daseq THEN             04442
                  BEGIN %restore any strings that are
                  suppressed%                                  04443
                  IF da.dasuppress THEN                        04444
                     BEGIN                                     04445
                     %restore display area if it is
                     suppressed%                               04446
                     resda(&da);                               04447
                     END                                       04448
                  ELSE                                         04449
                     BEGIN                                     04450
                     litreset ← TRUE;                          04451
                     dpset(dspjpf, endfil, endfil, endfil);
                                                               04452
                     daupdate(&da);                            04453
                     % the da.dalssup is set to 0 in daupdate
                     %                                         04454
                     END;                                      04455
                  END;                                         04456
               lplitreset ← FALSE;                             07140
               END                                             04457
            ELSE                                               07135
               BEGIN                                           07136
               lplitreset ← TRUE;                              07137
               lplitline ← savelitline;                        07138
               END;                                            07139
         track();                                              04458
         END;                                                  04459
= imlac0, = imlac1:                                            05817
         BEGIN                                                 05818
         IF littyflag := FALSE THEN                            05819
            BEGIN %should change to send char string also%     05820
            ttywindow(ttyda);                                  05821
            nlmode ← fulldisplay;                              05822
            END;                                               05823
         %restore suppressed file text display%                05824
            end ← $dpyarea + dal*dacnt;                        05825
            FOR &da ← $dpyarea UP dal UNTIL >= end DO          05826
               IF  da.daexis AND NOT da.daseq THEN             05827
                  BEGIN %restore any strings that are suppressed%
                                                               05828
                  %restore display area if it is suppressed% 05829
                     IF da.dasuppress THEN resda(&da);         05830
                  &ls ← da.dalsrt;                             05831
                  endls ← &ls + (da.dalssup := 0) * lsrtl;     05832
                     %dalssup points to the first string which is
                     not suppressed%                           05833
                  UNTIL &ls >= endls OR NOT ls.rtexis DO       05834
                     BEGIN                                     05835
                     *send* ← begmsg, 4+remfudge, remrsda,
                     da.dahandle.daidr1 + remfudge,
                     da.dahandle.daidr2 + remfudge, ls.rtlsid;
                                                               05836
                     !sout(dspjfn, chbmty+$send, -send.L);     05837
                     IF da.dalhandle THEN %this display linked to
```

```
                        another -- restore it's string also%     05838
                           !sout(dspjfn, chbmty+$send, -send.L);
                                                                 05839
                        &ls ← &ls + lsrtl;                       05840
                        END;                                     05841
                     END;                                        05842
            litapflag ← FALSE;                                   05843
            END;                                                 05844
         ENDCASE                                                 04460
            BEGIN                                                04461
            IF littyflag := FALSE THEN                           04462
               BEGIN                                             04463
               IF NOT SKIP !uasqd(#00000001B3 .V ttysim) THEN    04464
                  dismes(2, $"UASQD Failed in RSTLIT");          04465
               nlmode ← fulldisplay;                             04466
               END;                                              04467
            %restore suppressed file text display%               04468
               end ← $dpyarea + dal*dacnt;                       04469
               FOR &da ← $dpyarea UP dal UNTIL >= end DO         04470
                  IF  da.daexis AND NOT da.daseq THEN            04471
                     BEGIN %restore any strings that are suppressed%
                                                                 04472
                     %restore display area if it is suppressed% 04473
                        IF da.dasuppress THEN resda(&da);        04474
                     &ls ← da.dalsrt;                            04475
                     endls ← &ls + (da.dalssup := 0) * lsrtl;    04476
                        %dalssup points to the first string which is
                        not suppressed%                          04477
                     UNTIL &ls >= endls OR NOT ls.rtexis DO      04478
                        BEGIN                                    04479
                        rl.RH ← ls.rtlsid;                       04480
                        rl.LH ← da.dahandle;                     04481
                        IF NOT SKIP !JSYS rsda THEN              04482
                           dismes(2,$"rstlit RSDA JSYS error");  04483
                        IF da.dalhandle THEN %this display linked to
                        another -- restore it's string also%     04484
                           BEGIN                                 04485
                           rl.RH ← ls.rtlsid;                    04486
                           rl.LH ← da.dalhandle;                 04487
                           IF NOT SKIP !JSYS rsda THEN           04488
                              dismes(2,$"Link rstlit RSDA JSYS
                              error");                           04489
                           END;                                  04490
                        &ls ← &ls + lsrtl;                       04491
                        END;                                     04492
                     END;                                        04493
            litapflag ← FALSE;                                   04494
            END;                                                 04495
      litreset ← TRUE;                                           04496
      RETURN;                                                    04497
      END.                                                       04498
                                                                 04499
   (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%       01600
   LOCAL save, da;                                                 02191
   LOCAL STRING send[10];                                          05845
   REF da;                                                         02638
   CASE nldevice OF                                                05861
      = imlac0, = imlac1:                                          05862
         BEGIN                                                       05863
         *send* ← begmsg, 4+remfudge, remsdda, daid.daidr1 +
         remfudge, daid.daidr2 + remfudge, TRUE;                   05864
         !scout(dspjfn  chbmty+$send, -send.L);                    05865
         *send* ← begmsg, 3+remfudge, remrdda, daid.daidr1 +
         remfudge, daid.daidr2 + remfudge;                         05866
         !scout(dspjfn, chbmty+$send, -send.L);                    05867
         END;                                                        05868
      ENDCASE                                                      05869
         BEGIN                                                       05870
         IF NOT SKIP !sdda (daid, TRUE) THEN                       05871
            err($"SDDA Jsys error, CLRSEDA");                      05872
         IF NOT SKIP !rdda THEN                                    05873
            err($"RDDA Jsys error, CLRSEDA");                      05874
         END;                                                        05875
   RETURN(daid);                                                   02187
   END.                                                            02188
                                                                    02189
(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 $slittabs+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 ← 4407B8 .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);                                    06216
         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%                                         04123
   %--------------%                                                 04124
   LOCAL STRING send[10];                                           05876
   REF astrng;                                                      04125
   clearlit();                                                      04126
   supda(0); %suppress display of all text areas%                   04127
   setlit(); %set up lit area%                                      04128
   CASE nldevice OF                                                 04129
      = devlproc:                                                   04130
         BEGIN                                                      04131
         lpttywindow(litda.datop, litda.dabottom - litda.davinc);
                                                                    04132
         track();                                                   04133
         END;                                                       04134
      = imlac0, = imlac1:                                           05877
         BEGIN                                                      05878
         *send* ← begmsg, 4+remfudge, echda, 30B,
         litdahandle.daidr1 + remfudge, litdahandle.daidr2 +
         remfudge;                                                  05879
         !sout(dspjfn, chbmty+$send, -send.L);                      05880
         END;                                                       05881
      ENDCASE                                                       04135
         BEGIN                                                      04136
         IF NOT SKIP !uasqd(6C0000001B3 .V litdahandle) THEN        04137
            err($"UASQD Failed in LITTTY");                         04138
         END;                                                       0413
   littyflag ← TRUE;                                                04140
   nlmode ← typewriter;%should find some other way to do this%
                                                                    04141
   RETURN;                                                          04142
   END.

                                                                    04143
(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 litlop + (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);                                            04162
       %suppress display of any line segments above ycoord on the
       screen%                                                          04163
       %-------------%                                                  04164
       LOCAL end, da, ls, stop, dtop;                                   04165
       LOCAL STRING senf[20];                                           05882
       REF da, ls;                                                      04166
       end ← $dpyarea + dal*dacnt;                                      04167
       FOR &da ← $dpyarea UP dal UNTIL >= end DO                        04168
          IF da.daexis AND NOT da.daseq AND NOT da.dasuppress THEN
                                                                        04169
             IF (dtop ← da.datop) <= ycoord THEN                        04170
                BEGIN                                                   04171
                &ls ← da.dalsrt + da.dalssup * lsrtl;                   04172
                stop ←  da.dalsrt + da.dalsz * lsrtl;                   04173
                UNTIL &ls >= stop DO                                    04174
                   BEGIN                                                04175
                   IF ls.rtexis AND (dtop + ls.rty) <= ycoord THEN
                                                                        04176
                      BEGIN                                             04177
                      CASE nldevice OF                                  04178
                         = devlproc:                                    04179
                            BEGIN                                       04180
                            cline(da.daleft, ls.rty+da.datop,
```

```
                        lpxmax);                                        04181
                        END;                                            04182
                    = imlac0, = imlac1:                                 05883
                        BEGIN                                           05884
                        *send* ← begmsg, 5+remfudge, remssda,
                        da.dahandle.daidr1 + remfudge,
                        da.dahandle.daidr2 + remfudge,
                        ls.rtlsid, 0;                                   05885
                        !sout(dspjfn, chbmty+$send, -send.L);
                                                                        05886
                        IF da.dalhandle THEN                            05887
                            BEGIN                                       05888
                            !sout(ldspjfn, chbmty+$send, -send.L);
                                                                        05889
                            END;                                        05890
                        END;                                            05891
                    ENDCASE                                             04183
                        BEGIN                                           04184
                        rl.RH ← ls.rtlsid;                              04185
                        rl.LH ← da.dahandle;                            04186
                        IF NOT SKIP !JSYS ssda THEN                     04187
                            err($"SSDA JSYS error, litsup");            04188
                        IF da.dalhandle THEN                            04189
                            BEGIN                                       04190
                            rl.RH ← ls.rtlsid;                          04191
                            rl.LH ← da.dalhandle;                       04192
                            IF NOT SKIP !JSYS ssda THEN                 04193
                                err($"SSDA JSYS error, litsup");
                                                                        04194
                            END;                                        04195
                        END;                                            04196
                    END                                                 04197
                ELSE EXIT;                                              04198
                &ls ← &ls + lsrtl;                                      04199
                END;                                                    04200
            da.dalssup ← (&ls-da.dalsrt)/lsrtl;                         04201
            END;                                                        04202
    RETURN;                                                             04203
    END.

                                                                        04204
%...Lesser support routines...%                                         01411
(cntrlgetchar) PROCEDURE; %GETCHAR with CONTROL FILE STUFF%               075
    %*****************************************************
    DO NOT CALL THIS ROUTINE DIRECTLY == CALL RAWCHR INSTEAD
    *****************************************************%               076
    % This routine is just like GETCHAR except that it is used
    whenever the CONTROL FILE RECORD or PLAYBACK is in effect.%           077
                                                                          078
    %NOTE:  If inpwatchjfn is non-zero the time of char request, time
    when got char, and 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.%                            079
```

```
%--------------------%                                                  080
LOCAL char, gottime, begtime;                                           081
rubabt ← TRUE; %allow command to be aborted%                            082
IF inpjfn THEN                                                          083
   BEGIN %read time(request)%                                           084
   r1 ← inpjfn;                                                         085
   !JSYS bin;                                                           086
   begtime.tmchr1 ← r2;                                                 087
   !JSYS bin;                                                           088
   begtime.tmchr2 ← r2;                                                 089
   !JSYS bin;                                                           090
   begtime.tmchr3 ← r2;                                                 091
   !JSYS bin;                                                           092
   begtime.tmchr4 ← r2;                                                 093
   !JSYS bin;                                                           094
   begtime.tmchr5 ← r2;                                                 095
   !JSYS bin;                                                           096
   begtime.tmchr6 ← r2;                                                 097
   END                                                                  098
ELSE                                                                    099
   BEGIN                                                               0100
   !JSYS time;                                                         0101
   begtime ← r1;                                                       0102
   END;                                                                0103
IF inpwatchjfn THEN                                                    0104
   BEGIN %write out time(request)%                                     0105
   r1 ← inpwatchjfn;                                                   0106
   r2 ← begtime.tmchr1;                                                0107
   !JSYS bout;                                                         0108
   r2 ← begtime.tmchr2;                                                0109
   !JSYS bout;                                                         0110
   r2 ← begtime.tmchr3;                                                0111
   !JSYS bout;                                                         0112
   r2 ← begtime.tmchr4;                                                0113
   !JSYS bout;                                                         0114
   r2 ← begtime.tmchr5;                                                0115
   !JSYS bout;                                                         0116
   r2 ← begtime.tmchr6;                                                0117
   !JSYS bout;                                                         0118
   END;                                                                0119
IF nlcron AND nlstyp IN [0,2] THEN nlcrms(TRUE);                       0140
IF inpjfn THEN                                                         0120
   BEGIN %sleep for appropriate time%                                  0121
   %read time(gotchar)%                                                0122
      r1 ← inpjfn;                                                     0123
      !JSYS bin;                                                       0124
      gottime.tmchr1 ← r2;                                             0125
      !JSYS bin;                                                       0126
      gottime.tmchr2 ← r2;                                             0127
      !JSYS bin;                                                       0128
      gottime.tmchr3 ← r2;                                             0129
      !JSYS bin;                                                       0130
      gottime.tmchr4 ← r2;                                             0131
      !JSYS bin;                                                       0132
      gottime.tmchr5 ← r2;                                             0133
      !JSYS bin;                                                       0134
```

```
      gottime.tmchr6 ← r2;                                          0135
   %dismis for simulated char wait (time(gotchar) -
   time(requet))%                                                   0136
      r1 ← IF symtflg THEN gottime - begtime ELSE 0;               07144
      !JSYS simch;                                                 07145
      %IF symtflg THEN                                             07141
         BEGIN                                                     07142
         r1 ← gottime - begtime;                                    0137
         !JSYS simch;                                               0138
         END;%                                                     07143
   END;                                                             0139
%get next character%                                                0141
   r1 ← IF inpjfn THEN inpjfn ELSE ctty;                            0142
   !JSYS bin;                                                       0143
   char ← r2;                                                       0144
IF nlcron AND nlstyp IN [0,2] THEN nlcrms(FALSE);                   0145
%write out if being watched%                                        0146
   IF inpwatchjfn THEN                                              0147
      BEGIN                                                         0148
      %write time(gotchar)%                                         0149
         IF NOT inpjfn THEN                                         0150
            BEGIN                                                   0151
            !JSYS time;                                             0152
            gottime ← r1;                                           0153
            END;                                                    0154
         r1 ← inpwatchjfn;                                          0155
         r2 ← gottime.tmchr1;                                       0156
         !JSYS bout;                                                0157
         r2 ← gottime.tmchr2;                                       0158
         !JSYS bout;                                                0159
         r2 ← gottime.tmchr3;                                       0160
         !JSYS bout;                                                0161
         r2 ← gottime.tmchr4;                                       0162
         !JSYS bout;                                                0163
         r2 ← gottime.tmchr5;                                       0164
         !JSYS bout;                                                0165
         r2 ← gottime.tmchr6;                                       0166
         !JSYS bout;                                                0167
      %write char%                                                  0168
         r2 ← char;                                                 0169
         !JSYS bout;                                                0170
      END;                                                          0171
inptrf ← inpstp ← rubabt ← FALSE;                                   0172
RETURN(char);                                                       0173
END.                                                                0174
(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[1] ← $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 (cnar); %process view specs%                         07061
   LOCAL da, save, frmt;                                                07062
   REF da;                                                              07063
   LOCAL STRING locstr[20];                                             07064
   IF inhelp THEN RETURN;                                               07065
   CASE char OF                                                         07066
      = 'F :  %recreate the display using dafrmt%                       07067
        BEGIN                                                           07068
        &da ← lda();                                                    07069
        IF sysvspec.vsrlev THEN                                         07070
           sysvspec ← <SEQGEN, reslev>(sysvspec, getlev(da.dacsp));
                                                                        07071
        da.dapvs ← da.davspec ← cspvs ← sysvspec;                       07072
        da.dapvs2 ← da.davspc2 ← cspvs[1] ← sysvspec[1];                07073
        save ← da.davspec.vsdaft := TRUE;                               07074
        dafrmt (&da, 0);                                                07075
        da.davspec.vsdaft ← save;                                       07076
        END                                                             07082
      = 'f :  %recreate the display using daupdate%                     07083
        BEGIN                                                           07084
        &da ← lda();                                                    07085
        IF sysvspec.vsrlev THEN                                         07086
           sysvspec ← <SEQGEN, reslev>(sysvspec, getlev(da.dacsp));
                                                                        07087
        da.davspec ← cspvs ← sysvspec;                                  07088
        da.davspc2 ← cspvs[1] ← sysvspec[1];                            07089
        save ← da.davspec.vsdaft := TRUE;                               07090
        dpset(dspjpf, endfil, endfil, endfil);                          07091
        IF howformat(&da, da.dacsp.stfile, endfil : frmt) THEN          07092
           IF frmt THEN                                                 07093
              dafrmt(&da, 0)                                            07094
           ELSE                                                         07095
              daupdate (&da);                                           07096
        da.davspec.vsdaft ← save;                                       07097
        da.dapvs ← da.davspec;                                          07131
        da.dapvs2 ← da.davspc2;                                         07132
        END                                                             07103
      ENDCASE                                                           07104
        BEGIN                                                           07105
        sysvspec ← stklt(char, sysvspec, sysvspec[1] :
        sysvspec[1]);                                                   07106
```

```
            %call stklt to set viewspec%                                07107
         on($vspstr);                                                   07108
         dspvsp(sysvspec, sysvspec[1], 3);                              07109
         END;                                                           07110
   RETURN;                                                              07111
   END.
                                                                        07112
(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%                           0260
      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( 8"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);                                    0416
   %This procedure accepts a teletype numbr and a string, and types
   the string as a message on the tty.                            0417
      It additionally typeas a series of attention getting
      characters on the tty.%                                     0418
   REF astr;                                                      0419
   IF jdebug THEN RETURN; %Don't do it if debugging%              0420
```

```
        %Check that tty is available%                                   0421
           r1 ← getsbn($"TTYJOB");                                      0422
           !JSYS sysgt;                                                 0423
           IF (?R1.RH ← r2.RH) = 0 THEN RETURN; %Table not thee%        0424
           r1.LH ← ttyno;                                               0425
           IF (NOT SKIP !JSYS getab) OR (r1 # -1) THEN RETURN;          0426
        %First the attention getters%                                   0427
           r1 ← 4B5 .V ttyno;                                           0428
           r2← cnbptr(0) + $"
           ←←←←←--";                                                    0429
           r3 ← -8; %byte count%                                        0430
           !JSYS sout;                                                  0431
        %Now the string%                                                0432
           r1 ← 4B5 .V ttyno;                                           0433
           r2 ← cnbptr(0) + &astr;                                      0434
           r3 ← -astr.L;                                                0435
           !JSYS sout;                                                  0436
        %Now another EOL%                                               0437
           r1 ← 4B5 .V ttyno;                                           0438
           r2 ← EOL;                                                    0439
           !JSYS bout;                                                  0440
        RETURN                                                          0441
        END.                                                            0442
%...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
```

(MLK)IOEXEC (MLK)IOEXEC (MLK)IOEXEC (MLK)IOEXEC (MLK)IOEXEC (M

```
< NLS, IOEXEC.NLS;328, >, 10-OCT-74 11:51 KEV ;;;;

FILE ioexec   %  L10  <REL-NLS>IOEXEC  %% (L10,) (rel-nls,ioexec.rel,)
%
                                                                      02

% error handling %
                                                                    02519
   %                                                                02520
   Procedures that call err if anything wrong                       02521
      open, close, writefile, copfil, openall, closeall, rawopen    02522
   Procedures that can return FALSE with err message                02523
      sysopen, sysclose, opnfil, lgetjfn, sgtjfn, openpc, delpc,
      clsfil, rdhdr                                                 02524
   Procedures that can return warning message                       02525
      rdhdr, opnfil, delpc                                          02526
   %
                                                                    02527
%.....Declarations.....% % These should be moved to appropriate files:
utilty, data, etc.%
                                                                      03
   DECLARE                                                            05
      pc = 1, bfile = 4,                                            02826
      sub=1, suc=0,                                                   06
      dummy = 0, %for compiler bug%                                 02951
      gtjwrt = 4B5, gtjred = 1B5, gtjigdel = 1B3,                     09
      gtjcpc = 2B3, sgtjff = 1B6, jfnstf = 111100400001B;           010
   (lock) RECORD %lock word in file descriptor block%               01404
      lkinit[21], lkdirn[9], acctyp[3], ojdelf[1];                  01405
   REGISTER r1=1, r2=2, r3=3, r4=4;                                   019
                                                                      021
   REF tda;                                                           022
                                                                      023
%.....file opening.....%
                                                                     0440
   (open)   PROCEDURE % Open file routine.  If the value of jfn passed
   to this procedure is zero, it tries to get a jfn (using short get
   jfn with the file name contained in the string whose address is
   passed as the second parameter.)  If it cannot get the jfn, err is
   called.  The full file name is put into astrng.  If not already
   open, the original file whose jfn is passed or has been obtained is
   opened along with its partial copy (if it exists); the file is
   entered in the file status table and the NLS file number returned.
   If the file is already open, the old file number is returned.  If
   the file cannot be successfully opened, the file is removed from the
   file status table and err is called %                             0442
      (jfn,  % jfn of the original file to be opened; if zero,      03882
         we use the file name contained in astrng as parameter to
         sgtjfn to obtain a jfn %                                   03885
      astrng % Address of string containing file name (if jfn       03884
         zero); may be empty if jfn is a positive number.  In both
         cases the full file name will be placed in the string by
         jfnstr. %                                                  03887
      ):                                                            03886
      %----------------%                                             0452
      LOCAL fileno, fl, flgbits;                                     0453
      LOCAL STRING errs[150];                                       02282
```

```
    REF fl, astrng;                                                    0454
    %get full file name%                                               0455
    IF NOT jfn THEN                                                    0456
       BEGIN                                                           0457
       flgbits ← getgtjflg(write, origff, oldvrsn);                    0458
          %gets highest extant version or creates version 1%           0459
       IF NOT jfn ← sgtjfn(flgbits, &astrng, $errs) THEN               0461
          err(&errs);                                                  0464
       END;                                                            0467
    jfnstr(jfn, &astrng);                                              0468
    IF (fileno ← filnum(&astrng)) IN [1,filcnt] THEN                   0469
       BEGIN %already open--check validity%                            0470
       &fl ← flntadr(fileno);                                          0471
       IF fl.florig # jfn THEN                                        01623
          BEGIN                                                       01624
          rl ← jfn;                                                   01625
          IF NOT SKIP !JSYS closf THEN reljfn(jfn);                   01626
          jfn ← fl.florig;                                            01630
          END;                                                        01631
       IF (fl.flpart OR fl.florig) AND                                 0472
          fl.flhead THEN                                               0473
             RETURN(fileno, FALSE) % old file return %                 0474
       ELSE delfil(fileno);                                            0475
       END;                                                            0476
    fileno ← addfil(jfn, &astrng);                                     0477
    IF NOT opnfil(fileno, $errs) THEN                                  0478
       BEGIN                                                           01525
       delfil(fileno);                                                 03726
       err($errs);                                                     03727
       END;                                                            01527
    IF errs.L > empty THEN                                             03822
       BEGIN                                                           03823
       dismes (2, $errs);                                              03824
       END;                                                            03826
    RETURN(fileno, TRUE);  % new file %                                01519
    END.                                                               0481
                                                                       0482
(openwk) PROCEDURE % opens and resets work file specified by astr:
creates it with a default extension of .NLS if it does not exist.
It returns the stid of the origin statement of the file %              02474
    (jfn,  % Any value passed here is ultimately not used!!!           03891
       (The space is passed to opwknr which then does a lgetjfn
       anyway %                                                        03894
    astr   % The address of a string containing the name of the        03892
       work file to be opened. %                                       03895
    );                                                                 03893
    LOCAL fileno, stid;                                                02477
    REF astr;                                                          03896
    % open work file without resetting file without resetting. %       03563
       fileno ← opwknr(jfn, &astr);                                    03562
    % reset file%                                                      03564
       resetf(fileno);                                                 02491
    % get stid of origin %                                            03897
       stid ← 0;  %to clear all fields%                                03565
       stid.stfile ← fileno;                                           02492
       stid.stpsid ← origin;                                           03898
```

```
   RETURN(stid) END.                                                    02494

(opwknr) PROCEDURE % Opens (or creates) without resetting work file
specified in astr.  Returns the NLS fileno.  The file cannot be
already open!  (opwk handles this case; the two should be combined.)
%                                                                       03548
   (jfn,  % The space is used, but the value passed is not!!! %         03899
   astr   % The address of string containing the name of the            03901
      work file to be opened. %                                         03903
   );                                                                   03902
   LOCAL fileno, stid, fl;                                              03550
   LOCAL STRING fns[50], errsr[150];                                    03551
   REF astr, fl;                                                        03552
   *fns* ← *astr*; %get some working room%                              03553
   IF NOT (jfn ← lgetjfn (0, $fns, $nlsext, getgtjflg (write,
   origff, oldvrsn), $errsr)) THEN err ($errsr);                        03554
   jfnstr(jfn, $fns);                                                   03555
   fileno ← addfile(jfn, $fns);                                         03556
   %now do dummy open and close to create it if it does not exist%
                                                                        03557
      IF NOT sysopen(jfn, write, random, $errsr) THEN err($errsr);
                                                                        03558
      IF NOT sysclose(jfn .V 4B11, $errsr) THEN err($errsr);            03559
   IF NOT opnfil(fileno, $errsr) THEN err($errsr);                      03560
   IF errsr.L > empty THEN                                              03827
      BEGIN                                                             03828
      dismes (2, $errsr);                                               03829
      END;                                                              03831
   RETURN(fileno) END.
                                                                        03561

(opwk)   PROCEDURE % The same as opwknr, but permits a file to
already be open!  (accomplished through code which exists in open.
Should be combined with opwknr with parameter added to check if this
code need be executed.) %                                               03582
   (jfn,  % The space is used, but the value passed is not!!! %         03957
   astr   % The address of string containing the name of the            03958
      work file to be opened. %                                         03959
   );                                                                   03960
   LOCAL fileno, stid, fl;                                              03584
   LOCAL STRING fns[50], errsr[150];                                    03585
   REF astr, fl;                                                        03586
   *fns* ← *astr*; %get some working room%                              03587
   IF NOT (jfn ← lgetjfn (0, $fns, $nlsext, getgtjflg (write,
   origff, oldvrsn), $errsr)) THEN err ($errsr);                        03588
   jfnstr(jfn, $fns);                                                   03589
   IF (fileno ← filnum($fns)) IN [1,filcnt] THEN                        03590
      BEGIN %already open--check validity%                              03591
      &fl ← flntadr(fileno);                                            03592
      IF fl.florig # jfn THEN                                           03593
         BEGIN                                                          03594
         rl ← jfn;                                                      03595
         IF NOT SKIP !JSYS closf THEN reljfn(jfn);                      03596
         jfn ← fl.florig;                                               03597
         END;                                                           03598
      IF (fl.flpart OR fl.florig) AND                                   03599
         fl.flhead THEN                                                 03600
```

```
            RETURN(fileno)                                              03601
      ELSE                                                              03602
         BEGIN                                                          03603
         delfil(fileno);                                                03604
         RETURN(FALSE);                                                 03605
         END;                                                           03606
      END;                                                              03607
   fileno ← addfile(jfn, $fns);                                         03608
   %now do dummy open and close to create it if it does not exist%
                                                                        03609
      IF NOT sysopen(jfn, write, random, $errsr) THEN err($errsr);
                                                                        03610
      IF NOT sysclose(jfn .V 4B11, $errsr) THEN err($errsr);            03611
   IF NOT opnfil(fileno, $errsr) THEN err($errsr);                      03612
   IF errsr.L > empty THEN                                              03832
      BEGIN                                                             03833
      dismes (2, $errsr);                                               03834
      END;                                                              03836
   RETURN(fileno) END.
                                                                        03613

(openlock)  PROCEDURE % Open and lock file indicated by astr after
waiting for it to be free.  (Used primarily for systems files in
Journal, Ident systems, etc.)  (Uses OPLOCK to set and reset
openlock lock flag--  this is the only flag used to channel locks by
everyone using the journal, ident, and measurement systems(!!!))
Expunges directory in effort to try to get file opened.  Also does
some other wierd things.  This is a very poorly coded procedure
which is used in many important subsystems.  Returns stid of origin
statement of file.  SIGNALS or calls err if something wrong. %   03729
   (jfn, % NOT USED!!!!! %                                              03888
   astr  % Address of string containing name of file to be opened. %
                                                                        03889
   );                                                                   03890
   LOCAL stid,  times, mssflg, expunge, njfn, fileno, fl, jdirnm;
                                                                        03733
   LOCAL STRING fns[50];                                                03734
   REF astr, fl;                                                        03735
   stid ← 0;                                                            03743
   %Now check the lock on the file%                                     03744
      mssflg ← times ← expunge ← 0;                                     03745
      LOOP BEGIN                                                        03746
         *fns* ← *astr*;                                                03747
         oplock(1); %mke sure only we are locking a file%               03748
         IF (fileno ← rawopen($fns, 0)) THEN EXIT; %Got it%             03749
         oplock(0);                                                     03750
         IF jdebug THEN BEGIN                                           03751
            crlf();                                                     03752
            typeas($fns);                                               03753
            crlf();                                                     03754
            typeas($sar);                                               03755
            END;                                                        03756
         IF NOT expunge THEN                                            03757
            BEGIN                                                       03758
            expunge ← TRUE;                                             03759
            %expunge connected directory%                               03760
            !JSYS GJINF;                                                03761
```

```
            r1 ← r2;                                                    03762
            !JSYS deldf;                                                03763
            END;                                                        03764
         IF (times ← times+1) > 24 THEN  %too long%                     03765
            BEGIN                                                       03766
            dolock(); %check for a file error from slinker%             03767
            SIGNAL(2,$"  File Busy--Try Again Later");                  03768
            END;                                                        03769
         IF FALSE %NOT mssflg% THEN BEGIN                               03770
            crlf(); typeas($"File locked: ");                           03771
            typeas(&astr);                                              03772
            typeas($"--Waiting");                                       03773
            BUMP mssflg END;                                            03774
         %Now Wait for 2500 ms%                                         03775
            r1 ← 2500; !JSYS dismS; %167B%                              03776
         END;                                                           03777
      %By here, file is ours%                                           03778
         IF NOT (&fl ← flntadr(fileno)).flpart THEN                     03779
            BEGIN %make a partial copy%                                 03780
            !JSYS gjinf;                                                03781
            jdirnm ← r2; %Put it in the connected directory for this
            one%                                                        03782
            IF NOT setfdb(jdirnm, cinit, &fl) THEN                      03783
               BEGIN                                                    03784
               *lit* ← "Cannot Write File";                             03785
               GOTO oplockfail;                                         03786
               END;                                                     03787
            %File is now locked--get PC name and make pc%               03788
               IF fl.flpcst THEN freestring( fl.flpcst := 0,
               $dspblk);                                                06675
               fl.flpcst ← cpcnam(fl.flastr, jdirnm);                   03789
               IF NOT makepc(&fl, fileno, $lit) THEN                    03790
                  BEGIN                                                 03791
                  %expunge connected directory%                         03792
                     !JSYS gjinf;                                       03793
                     r1 ← r2;                                           03794
                     !JSYS deldf;                                       03795
                  IF NOT makepc(&fl, fileno, $lit) THEN                 03796
                     BEGIN                                              03797
                     setfdb(0, 0, &fl); %Unlock it%                     03798
                     (oplockfail):                                      03799
                     *lit* ← "Openlock Fail: ", *fns*, EOL, *lit*;
                                                                        03800
                     oplock(0); %reset flag to let others lock
                     files%                                             03801
                     err($lit);                                         03802
                     END;                                               03803
                  END;                                                  03804
            END;                                                        03805
      oplock(0); %reset flag to let others lock files%                  03806
   IF mssflg THEN crlf();                                               03807
   stid.stfile ← fileno;                                                03808
   stid.stpsid ← origin;                                                03809
   RETURN(stid) END.
                                                                        03817
(opnctfile) PROCEDURE % open initial file (in any directory)
```

```
specified by the strings whose addresses are passed in initstr and
userstr. CALLED ONLY IN (JNLDEL, OLDIST).  Return stid of origin if
opened, and FALSE if not.  In addition to opening it, it also locks
it by creating a partial copy.  Enables to permit the operator
through this procedure to get to the initial file of any user.
Calls rawopen. %                                                    02671
   (initstr,  % Address of string containing initials of user. %
                                                                    03904
   userstr  % Address of string containing user name %             03905
   );                                                               03906
   % Note that there is a potential timing problem in that we don't
   have control over the file if it is not locked when we first open
   it.  We should fix this when we decide how to fix the problem for
   NLS in general %                                                 02673
   LOCAL pcap, fileno, ctlstid, fl, dirnm, lcinit, updinitf;        02674
   LOCAL STRING fname[50], ucustr[50], errstr[200];                 02675
   REF initstr, userstr, fl;                                        02676
   %Initialise parms, and set up signal%                            02677
      fileno ← ctlstid ← updinitf ← 0;                              02678
      ctlstid.stpsid ← origin;                                      02679
      ON SIGNAL ELSE                                                02680
         BEGIN                                                      02681
         close(fileno := 0);                                        02682
         RETURN(FALSE);                                             02683
         END;                                                       02684
   %Build file name%                                                02685
      *fname* ← '<, *userstr*, '>, *initstr*, ".NLS";               02686
   %enable wheel/operator%                                          02687
      pcap ← enablw();                                              02688
   ctlstid.strile ← fileno ← rawopen($fname, 0);                    03317
   %Disable wheel/operator%                                         02690
      IF pcap # -1 THEN disablw(pcap);                              02691
   IF fileno = 0 THEN                                               02689
      BEGIN                                                         03319
      IF sar.L < 200 THEN BEGIN                                     03508
         *errstr* ← *sar*;                                          03513
         *sar* ← *fname*, ": ", *errstr*;                           03514
         END;                                                       03515
      specttyout(oprtty, $sar); %tell operator about control file
      problem%                                                      03321
      RETURN(FALSE);                                                03318
      END;                                                          03320
   &fl ← fintadr(fileno);                                           02692
   %get users directory number%                                     03290
      *ucustr* ← *userstr* , 0;                                     03292
      astruc($ucustr);                                              03293
      r2 ← chbptr(0) + $ucustr;                                     03284
      r1 ← 1;                                                       03285
      !JSYS stdir;                                                  03286
      !JRST opncterr;                                               03287
      !JRST opncterr;                                               03289
      dirnm ← r1.RH;                                                03288
      lcinit ← setcinit(&initstr); %make 5-bit ident%               03283
   IF NOT fl.flpart THEN                                            02693
      BEGIN %create Partial Copy%                                   02694
      pcap ← enablw();                                              02695
```

```
      setfdb(dirnm, lcinit, &fl);                                    02698
      IF pcap # -1 THEN disablw(pcap);                               02699
      %file is locked...create PC%                                   02700
         IF fl.flpcst THEN freestring( fl.flpcst := 0, $dspblk);
                                                                     06676
         fl.flpcst ← cpcnam(fl.flastr, dirnm);                       02701
         IF NOT makepc(&fl, fileno, $lit) THEN                       02702
            BEGIN                                                    02703
            pcap ← enablw();                                         02704
            setfdb(0, 0, &fl);                                       02705
            (opncterr):                                              03291
            IF pcap # -1 THEN disablw(pcap);                         02706
            close(fileno:= 0);                                       02707
            RETURN(FALSE);                                           02708
            END;                                                     02709
      END                                                            02710
   ELSE                                                              03295
      BEGIN %see if locked by this user%                             03302
      r1 ← fl.florig;                                                03296
      r2 ← 1000024B;                                                 03297
      r3 ← $r3;                                                      03298
      !JSYS gtfdb;                                                   03299
      IF r3.lkinit # lcinit OR r3.lkdirn # dirnm THEN updinitf ←
      TRUE;                                                          03300
      END;                                                           03301
   RETURN(ctlstid, updinitf);                                        02711
   END.
                                                                     02712
(openall)   PROCEDURE;  % open all files in file status table.
Returns FALSE if cannot open one of the files in the file list--
used when user continues from EXEC after quit (files were closed but
not deleted from filst) %                                             0493
   %--------------------%                                           01783
   LOCAL fileno, count, fl;                                           0497
   REF fl;                                                            0498
   IF filcnt NOT IN /1,filmax/ THEN RETURN(FALSE);                    0499
   fileno ← 1;                                                        0500
   count ← 0;                                                       01518
   &fl ← $filst;                                                      0501
   DO BEGIN                                                           0502
      IF fl.flexis THEN                                               0503
         IF NOT opnfil(fileno, $lit) THEN                           02283
            delfil(fileno)                                          01513
         ELSE                                                       01514
            BEGIN                                                   01515
            BUMP count;                                             01516
            END;                                                    01517
      &fl ← &fl + filstl;                                             0506
      END                                                             0507
   UNTIL (fileno ← fileno + 1) > filcnt;                              0508
   RETURN(count);                                                     0510
   END.                                                               0511
                                                                      0512
(opnfil) PROCEDURE % Primitive open file routine-- read/write
access.  The original file and the partial copy for the given file
number are opened by this routine.  Does not allow pc to be opened
```

```
for read unless sysopen fails on pc, (disk allocation exceeded, for
example).  Returns FALSE if original file cannot be opened.  See
coropnfil. %                                                        03708
   (fileno, % NLS file number of file to be opened. %               03912
   astrng   % Address of string to be used for error messages by
                                                                    03914
      coropnfil and the procedures it calls; is emptied by
      coropnfil, then filled if error (or other information) is
      passed. %                                                     03916
   ):                                                               03915
   %-----------------%                                             03714
   REF astrng;                                                      03716
   RETURN( coropnfil(fileno, &astrng, readwrite) );                 03723
   END.                                                             03724
                                                                    03725
(coropnfil) PROCEDURE % Primitive open file-- specified access. %
                                                                    03644
   % Open file specified by NLS file number passed as the first
   parameter and its partial copy.  The access of the partial copy
   may be forced by specifying read access as the third parameter.
   Astrng is used for error and information strings.  It is emptied
   at the beginning of this routine.  Error return is FALSE.  It is
   the responsibility of the calling procedure to check the value
   and decide upon a course of action.  Note that there may be
   messages even if a TRUE return is made (something about the
   status of the PC, for example).  Thus it is also recommended that
   the calling procedure check the length of the string and display
   it if necessary.  THERE IS NO USER INTERACTION (I.E., MESSAGE
   PRINT OUT) IN THIS PROCEDURE. %                                  07032
   (fileno,  % NLS file number of file to be opened. %              03917
   astrng,   % Address of string to be used for error and           03918
      information. %                                                03922
   accesz    % Access of the partial copy desired. (read,           03919
      readwrite, or write) %                                        03921
   );                                                               03920
   %accesz refers to PC and may be read%                            03646
   LOCAL fl, flags, version, jobno, dirno, fdbwd;                   03647
   LOCAL STRING intstr[5], auxerr[150];                             03648
   REF astrng, fl;                                                  03649
   acchecked ← FALSE;                                               07033
   &fl ← flntadr(fileno);                                           03650
   IF fileno NOT IN [1,filcnt] OR                                   03651
      filcnt NOT IN [1,filmax] OR                                   03652
      NOT fl.flexis THEN                                            03653
         BEGIN                                                      03654
         *astrng* ←"NLS system error";                              03655
         RETURN(FALSE);                                             03656
         END;                                                       03657
   %open files%                                                     03658
      %get jfn for original file%                                   03659
      IF NOT fl.florig THEN                                         03660
         BEGIN                                                      03661
         *stn* ← *[fl.flastr]*;                                     03662
         flags ← getgtjflg(read, origff, dfltvrs);                  03663
         IF NOT fl.florig ← sgtjfn(flags, $stn, &astrng)            03664
         THEN RETURN(FALSE);                                        03665
```

```
      END;                                                          03666
   %open original file%                                             03667
   IF NOT sysopen(fl.florig, read, random, &astrng) THEN            03668
      RETURN(FALSE);                                                03669
   %get jfn for partial copy, if exists; open pc%                   03670
   %get job and directory number of this user%                      03671
      !JSYS gjinf;                                                  03672
      dirno ← r1; %login directory%                                 03673
   %construct pc name%                                              03674
      IF fl.flpcst THEN freestring( fl.flpcst := 0, $dspblk);
                                                                    06677
      fl.flpcst ← cpcnam(fl.flastr, dirno);                         03675
   %get job and directory number for file lock%                     03676
      r1 ← fl.florig;                                               03677
      r2 ← 100002B; %2ND (octal) word in the file                   03678
      data block%                                                   03679
      r3 ← $r3; %put word in r3%                                    03680
      !JSYS gtfdb; %get file data block%                            03681
   *astrng* ← NULL;                                                 03820
   IF r3.lkinit = cinit AND r3.lkdirn = dirno THEN                  03682
      BEGIN                                                         03683
      openpc(fileno, &astrng, accesz);                              03684
      END                                                           03685
   ELSE                                                             03686
      BEGIN %perhaps old style initials%                            03687
      fdbwd ← r3;                                                   03688
      *intstr* ← NULL;                                              03689
      trnsint(cinit, $intstr);                                      03690
      IF fdbwd.lkinit .A 4B6 AND fdbwd.lkdirn = dirno               03691
      AND intstr.1 = 3 AND intstr[1].oldint = fdbwd.lkinit THEN
                                                                    03692
         BEGIN                                                      03693
         openpc(fileno, &astrng, accesz);                           03694
         END                                                        03695
      ELSE IF  fdbwd.lkdirn # 0 THEN %file locked by someone
      else%                                                         03696
         fl.fllock ← TRUE; %set lock bit in file table%             03697
      END;                                                          03698
IF NOT rdhdr(fileno, $auxerr) THEN                                  03699
   %rdhdr writes header address into filst entry for this file
   number--initializes the header if things are not cool%           03700
   BEGIN                                                            03701
   % Append rdhdr mesage to astrng %                                03702
   IF auxerr.L THEN *astrng* ← *astrng*, SP, *auxerr*;              03703
   RETURN(FALSE);                                                   03704
   END;                                                             03705
IF NOT acchecked THEN                                               07034
   BEGIN                                                            07035
   ON SIGNAL ELSE                                                   07036
      BEGIN                                                         07037
      IF sysmsg THEN *astrng* ← *astrng*, SP, *[sysmsg]*;           07038
      RETURN (FALSE);                                               07039
      END;                                                          07040
   IF NOT vrprvacc (fileno, 0, 0, $initsr) THEN                     07041
      err ($"Private file; access denied to you");                  07042
   END;                                                             07043
```

```
   RETURN(TRUE);                                                    03706
   END.
                                                                    03707
(rawopen) PROCEDURE % Primitive open at any cost-- Open the file
specified in astr, enter it into the file list.  Open the partial
copy whether the file is locked by another or not.  Used only for
special subsystem files (journal, etc.)  Return fileno if
successful, 0 otherwise %                                           01899
   (astr, % address of name string of file to be opened. %          03928
   killpc % If TRUE, may unlock file if anything wrong with PC.     03929
      If FALSE, and cannot open PC, do not unlock file, but call err
      %                                                             03931
   );                                                               03930
   LOCAL fileno, jfn, dirnum, fl, pcflag;                           01903
   REF astr, fl;                                                    01904
   fileno ← 0;                                                      01905
   IF (jfn ← sgtjfn(getgtjfg(write, origff, oldvrsn), &astr, $sar))
   = 0 THEN RETURN(0); %File not there%                             01906
   ON SIGNAL                                                        01907
      ELSE                                                          01949
         BEGIN                                                      01947
         IF fileno THEN delfil(fileno := 0)                         02856
         ELSE reljfn(jfn);                                          02347
         RETURN(0);                                                 01951
         END;                                                       01952
   jfnstr(jfn, &astr);                                              01908
   LOOP                                                             02646
      BEGIN                                                         02647
      IF (fileno ← filnum(&astr)) IN [1,filcnt] THEN                02632
         BEGIN %already open--check validity%                       02633
         &fl ← flntadr(fileno);                                     02634
         IF (fl.flpart OR fl.florig) AND fl.flhead THEN             02641
            BEGIN                                                   02652
            reljfn(jfn);                                            02653
            EXIT LOOP                                               02643
            END                                                     02654
         ELSE delfil(fileno);                                       02644
         END;                                                       02645
      &fl ← flntadr(fileno ← addfile(jfn, &astr));                  01909
      IF NOT sysopen(jfn, read, random, $sar) THEN err($sar);       01910
      EXIT LOOP;                                                    02655
      END;                                                          02648
   ON SIGNAL ELSE BEGIN close(fileno); RETURN(0) END;               01911
   r1 ← fl.florig; r2 ← 1000024B; r3 ← $r3; !JSYS gtfdb;            01912
   IF r3.lkdirn # 0 THEN                                            01913
      dirnum ← r3.lkdirn                                            01953
   ELSE                                                             01954
      BEGIN                                                         01955
      !JSYS gjinf;                                                  01914
      dirnum ← r1                                                   01956
      END;                                                          01957
   IF fl.flpcst THEN freestring( fl.flpcst := 0, $dspblk);          06678
   fl.flpcst ← cpcnam(fl.flastr, dirnum);                           01915
   IF NOT openpc(fileno,  sar, readwrite) THEN                      01916
      IF killpc THEN                                                01959
         IF maywrt(fileno) THEN                                     02297
```

```
            BEGIN %something wrong with pc%                              01958
            %unlock the file%                                            01917
            r1.LH ← 2^B; r1.RH ← jfn;                                    01918
            r2 ← 0; r2.1kdirn ← r2.1kinit ← 36M;                         01919
            r3 ← 0;                                                      01920
            !JSYS chfdb;                                                 01921
            dismes(1, $"File Unlocked");                                 01960
            END                                                          01922
         ELSE NULL                                                       02298
      ELSE err($sar);                                                    01923
   rdhdr(fileno, $sar);                                                  01924
   RETURN(fileno) END.                                                   01925

(openpc)  PROCEDURE % Opens the PC for the file in fileno if it
exists with access specified in acc (if possible).  Returns True if
file does not have  PC or PC opened ok. Returns False if PC open
fail, astr contains message.  %                                          01872
   (fileno, % Of the orignal file whose PC is to be opened. %            03923
   astr,    % Address of astr in which error (or information)           03924
      string will be placed. %                                           03927
   acc      % Access desired for PC. %                                   03925
   );                                                                    03926
   LOCAL fl, dirnum;                                                     01878
   REF astr, fl;                                                         01879
   &fl ← flntadr(fileno);                                                01880
   r1 ← fl.florig; r2 ← 1000024B; r3 ← $r3;                              01881
   !JSYS gtfdb;                                                          01882
   IF (dirnum ← r3.1kdirn) = 0 THEN RETURN(TRUE);                        01883
   IF fl.flpcst = 0 THEN                                                 01884
      BEGIN                                                              01944
      *astr* ← "No PC Name String";                                      01885
      RETURN(FALSE);                                                     01886
      END;                                                               01945
   %Now get jfn%                                                         01887
      IF NOT fl.flpart ← sgtjfn(getgtjflg(read, chkpcf, dfltvrs),
      fl.flpcst, &astr) THEN                                             01888
         BEGIN                                                           02423
         *astr* ← "PC Does Not Exist";                                   02424
         filepart[fileno] ← FALSE;                                       03311
         RETURN(FALSE)                                                   02425
         END                                                             02426
      ELSE filepart[fileno] ← TRUE;                                      03310
   %Now try to open #t%                                                  01891
      IF acc = read  OR NOT sysopen(fl.flpart, readwrite, random,
      &astr) THEN                                                        01892
         BEGIN                                                           01942
         IF sysopen(fl.flpart, read, random, &astr) THEN                 03566
            BEGIN %PC open for read only-- disc allocation exceeded.
            etc.%                                                        03567
            fl.flpcread ← TRUE;                                          03574
            *astr* ← "File read only: ", *astr*;                         03573
            END                                                          03569
         ELSE %Cannot open PC at all%                                    03570
            BEGIN                                                        03571
            reljfn(fl.flpart := 0);                                      01893
            filepart[fileno] ← FALSE;                                    03312
```

```
            *astr* ← "PC Open Fail", EOL, *astr*;                    02427
            END;                                                     03572
         RETURN(FALSE)                                               01896
         END;                                                        01943
   RETURN(TRUE);                                                     01897
   END.
                                                                     01898
%.....file closing.....%
                                                                     03907
   (close)   PROCEDURE % Close original file and the partial copy and
   delete from file status table.  Calls err if error occurs.  writeout
   called three times: explicitly, in clsfil, and in delfil.  Clean
   this up. %                                                         0483
      (fileno % fileno of file to be closed %                        03932
      );                                                             03934
      %----------------%                                              0486
      %*****RETURN IF FILENO IS ZERO******%                          02669
         IF NOT fileno THEN RETURN;                                  02670
      % write out pages of file and PC and close them %              03938
         IF NOT writeout(fileno, $lit)
         OR NOT clsfil(fileno, $lit) THEN err($lit);                 02015
      delfil(fileno);                                                 0489
      RETURN;                                                         0490
      END.                                                            0491
                                                                      0492
   (closeu)  PROCEDURE % Close and update file %                      02657
      (fileno % fileno of file to be closed %                        03936
      );                                                             03937
      ON SIGNAL ELSE close(fileno := 0); %Close the file if update
      fails%                                                         02664
      IF fileno = 0 THEN RETURN;                                     02659
      updtfl(fileno, 1, 0);                                          02661
      %Now delete some past copies%                                  02662
         r1 ← [fintadr(fileno)].florig;                              02663
         r2 ← 3;                                                     02664
         IF NOT SKIP !JSYS delnf THEN NULL;                          02665
      close(fileno);                                                 02667
      RETURN END.
                                                                     02668
   (sigclose) PROCEDURE % Close file after SIGNAL.  Called in
   subsystems with NLS system files;  If journal file, lock journal
   system.  Close the file specified by fileno. %                    02713
      (fileno % fileno of file to be closed %                        03939
      );                                                             03940
      %This is a rather specialised procedure which is intended to be
      useful to submodes of NLS which use system-type NLS files.     02714
         It is presumed to be called from a SIGNAL, and takes the
         following action:                                           02715
            It checks to see if the value in sysgnl is -5 (badfile) or
            -6(I/O Data Error).                                      02716
               If it is, then it tries to determine whether or not the
               file involved is the one represented by fileno.       02717
               If it is, a message is typed onto the logging TTY, and
               the Journal is locked (insofar as at the time of writing
               all system files are used by the Journal--this should
               probably be changed later.                            02718
```

```
        The file is closed if fileno is non-zero.                    02719
   %                                                                 02720
   IF (sysgnl = -5) AND (fileno = bfilno) THEN                       02721
      BEGIN                                                          02722
      %This is the bad file%                                         02723
      lockjo(1); %lock Journal%                                      02724
      *lit* ← "Bad System File = ", *[[flntadr(fileno)],flastr]*;
                                                                     02725
      specttyout(0, $lit);                                           02726
      specttyout(30B, $lit);                                         02727
      bfilno ← 0;                                                    02728
      END;                                                           02729
   IF sysgnl = -6 THEN                                               02730
      BEGIN %I/O Data Error--see if this is the file%                02731
      %Code goes here to figure out which file it was%               02732
      END; %For now, simply pass it on%                              02733
   close(fileno);                                                    02734
   RETURN;                                                           02735
   END.
                                                                     02736
(closeall) PROCEDURE; % Close all files in file status table. %       0513
   %----------------------%                                          01784
   LOCAL fileno, fl;                                                  0517
   REF fl;                                                            0518
   IF filcnt NOT IN [0,filmax] THEN rerror();                         0519
   fileno ← 0;                                                        0520
   &fl ← $filst;                                                      0521
   UNTIL (fileno ← fileno + 1) > filcnt DO                            0522
      BEGIN                                                           0523
      IF fl.flexis AND NOT clsfil(fileno, $lit) THEN                  0524
         dismes(1, $lit);                                            01971
      &fl ← &fl + filstl;                                             0525
      END;                                                            0526
   RETURN;                                                            0528
   END.                                                               0529
                                                                      0530
(clsfil) PROCEDURE % Primitive close file routine. Does not
delete file from file status table, but does do writeout. The
original file and the partial copy are closed by this routine. If
anything goes wrong, it returns false with error message in astrng.
Close, closeu, closeall, and clsfil should be examined for possible
combination. %                                                        0571
   (fileno, % File number of file to be closed. %                   03941
   astrng   % Address of string to be used for error message. %     03942
   );                                                              03943
   %----------------%                                               0576
   LOCAL fl;                                                        0577
   REF fl, astrng;                                                  0578
   IF fileno NOT IN [1,filcnt] OR filcnt NOT IN [1,filmax]          0579
   THEN                                                             0580
      BEGIN                                                        01969
      *astrng* ← "NLS system error";                               01966
      RETURN(FALSE);                                               01967
      END;                                                         01968
   &fl ← flntadr(fileno);                                           0581
   IF NOT writeout(fileno, &astrng) THEN                           02016
```

```
      RETURN(FALSE);                                                    02017
   IF fl.flpart THEN                                                     0582
      BEGIN                                                              0583
      IF NOT sysclose(fl.flpart, &astrng) THEN                           0584
         RETURN(FALSE);                                                 01970
      fl.flpcread ← fl.flpart ← 0;                                      01622
      filepart[fileno] ← FALSE;                                         03309
      END;                                                               0605
   IF fl.florig AND NOT sysclose(fl.florig, &astrng) THEN                0591
      RETURN(FALSE);                                                    01785
   fl.florig ← 0;                                                        0598
   RETURN(TRUE);                                                         0617
   END.                                                                  0618
                                                                         0619
(closepc) PROCEDURE % Closes the PC for the file in fileno if it
exists. (Currently used only in ident system.) %                        03217
   (fileno % of file whose PC is to be closed %                         03944
   );                                                                   03945
   LOCAL fl, dirnum;                                                    03223
   LOCAL STRING work[100];                                              03263
   REF astr, fl;                                                        03224
   &fl ← flntadr(fileno);                                               03225
   IF fl.flpart THEN                                                    03235
      BEGIN                                                             03255
      IF NOT writecut(fileno, $work) THEN err($work);                   03256
      IF NOT sysclose(fl.flpart, $work) THEN                            03249
         reljfn(fl.flpart);                                             03251
      fl.flpcread ← fl.flpart ← 0;                                      03237
      filepart[fileno] ← FALSE;                                         03313
      IF NO  rdhdr(fileno, $work) THEN err($work);                      03262
      END;                                                              03239
   RETURN;                                                              03247
   END.
                                                                        03248
%.....null file, file reset, and initial file routines.....%
                                                                        01064
(openid) PROCEDURE;  % open initial file; constructs file name from
login directory name and current initials. creates the file if it
does not exist.  Returns file number. %                                 01066
   %----------------%                                                   01073
   LOCAL jfn, flgbits, word;                                            01074
   LOCAL STRING dname[30], nfname[50];                                  01739
   %build file name%                                                    01075
   %get login directory name%                                           01732
   gdname(cuno, $dname);                                                02324
   *nfname* ← '<, *dname*, '>, *initsr*, ".NLS";                        01076
   flgbits ← getgtjflg(write, origff, oldvrsn);                         01078
      %gets highest extant version or creates version 1%                01079
   IF NOT jfn ← sgtjfn(flgbits, $nfname, $lit) THEN err($lit);          01081
   ON SIGNAL                                                            02328
      =errsig:  % open has failed and called err %                      02329
         BEGIN                                                          02330
         ON SIGNAL ELSE;  % no longer catch signals %                   02333
         % open with write and close to create the file %               02339
            % if open fails, jfn will be released, must get it back
            %                                                           02355
```

```
        % will be initialized by rdhdr %                                02340
        IF NOT jfn ← sgtjfn(figbits, $nfname, $lit) THEN err($lit);
                                                                        02354
        word ← 0;                                                       06725
        word.fdbnar ← TRUE;          %say dont archive%                 06726
        chnfdb(jfn, 17B, word, word); % set file don't archive %
                                                                        06721
        IF NOT sysopen(jfn, write, random, $lit) OR
        NOT sysclose(jfn .V 4B11, $lit) THEN err($lit);                 02331
        % set up last read date of initial file %                       07080
           !gtfdb ( jfn, 1B6+$fdbref, $lstred);                         07081
        RETURN(open(jfn, $nfname));                                     02335
        END;                                                            02336
     ELSE;                                                              02337
   % set up last read date of initial file %                           07082
     !gtfdb ( jfn, 1B6+$fdbref, $lstred);                               07083
   RETURN(open(jfn, $nfname)) END.                                      02330
                                                                        01101
(openul1) PROCEDURE  % create file with given name.  Returns file
number. (Almost identical with openid except for name creation and
flags sent to sgtjfn; they should be combined (add flag parameter).)
%                                                                       03027
   (nfname % Address of name of file to be opened. %                    03946
   ):                                                                   03947
   %----------------%                                                   03029
   LOCAL jfn;                                                           03030
   IF NOT jfn ← sgtjfn(gtjoof, nfname, $lit) THEN err($lit);            03038
   ON SIGNAL                                                            03039
      =errsig:  % open has failed and called err %                      03040
         BEGIN                                                          03041
         ON SIGNAL ELSE;  % no longer catch signals %                   03047
         % open with write and close to create the file %               03042
            % if open fails, jfn will be released, must get it back
            %                                                           03043
         % will be initialized by rdhdr %                               03044
         IF NOT jfn ← sgtjfn(gtjoof, nfname, $lit) THEN err($lit);
                                                                        03045
         IF NOT sysopen(jfn, write, random, $lit) OR
         NOT sysclose(jfn .V 4B11, $lit) THEN err($lit);                03046
         RETURN(open(jfn, nfname));                                     03048
         END;                                                           03049
      ELSE;                                                             03050
   RETURN( open(jfn, nfname) );                                         03051
   END.                                                                 03053
                                                                        03052
(resetf) PROCEDURE % Reset file-- null partial file%                    01102
   (fileno % NLS file number of file to be reset. %                     03948
   ):                                                                   03949
   % This routine is called to convert the file whose number            01103
   is passed to a null file.  The original file is unchanged            01104
   but the PC is zapped.%                                               01105
   %----------------%                                                   01106
   IF fileno NOT IN [1, filcnt] THEN err($"NLS system error");          01109
   IF NOT writeout(fileno, $lit)
   OR NOT delpc(fileno, $lit) THEN err($lit);                           02025
   IF lit.L > empty THEN                                                01999
```

```
      BEGIN                                                          02572
      dismes(2, $lit);                                               02573
      END;                                                           02575
   intfil(fileno);                                                   01112
   RETURN;                                                           01117
   END.                                                              01118
                                                                     01119
%.....file initialization.....%
                                                                      0976
(intfil) PROCEDURE  % Initialize file-- initialize header and create
dummy origin statement %                                              0978
   (fileno % NLS file number of file being initialized %             03950
   );                                                                03951
   %------------------%                                              0981
   LOCAL stid, ring;                                                  0982
   LOCAL TEXT POINTER orgpoint;                                      03961
   REF ring;                                                          0983
   % Initialize file header, get origin ring & stid %                03962
      setfil(fileno);                                                 0984
      stid ← newrng(fileno : &ring);                                  0985
   % Check validity-- should be origin %                             03963
      IF stid.stpsid NOT= origin THEN err($"NLS system error");       0986
   % Fix up ring flags for origin %                                  03964
      ring.rhf ← TRUE;                                                0987
      orgpoint ← orgpoint[1] ← 0;                                     0988
      orgpoint.stpsid ← ring.rsuc ← ring.rsub ← origin;               0989
      orgpoint.stfile ← fileno;                                       0990
   % Set text of origin statement %                                  03965
      gdmys($orgpoint);                                               0991
   RETURN;                                                            0992
   END.                                                               0993
                                                                      0994
(setfil) PROCEDURE  % Initialize file header and core page status
table.  The header of the file whose number is passed is initialized
by this routine.  All status blocks are set to empty, the marker
table is set to empty, and the CORPST(core-page-status-table)
entries pertaining to this file (other than the header) are set to
indicate that no pages are loaded. %                                  03448
   (fileno % NLS file number of file being initialized %             03952
   );                                                                03953
   %--------------------%                                            03455
   LOCAL  index, header, pgindex, fb, fl, block;                     03456
   REF fb, fl;                                                       03457
   %initialize file header by updating dummy file header             03458
   and copying that to the real file header%                         03459
      nlsvwd ← 1;                                                    03460
      namdl1 ← dfnmdl; %use default name delimiters from user-option
      page%                                                          03461
      namdl2 ← dfnmdr;%use default name delimiters from user-option
      page%                                                          03462
      % file owner (user number) %                                  03966
         !JSYS gjinf;                                                03463
         funo ← r2;                                                  03464
      finit ← lwtim ← sidcnt ← 0;                                    03465
      % creation date %                                             03967
         !JSYS gtad;                                                 03466
```

```
        fcredt ← r1;                                                    03467
      mkrtbl ← 0;                                                       03968
      % clear ring and data status tables in header %                   03969
        clrngs($rngst);                                                 03469
        cldtbs($dtbst);                                                 03470
    &fl ← fintadr(fileno);                                              03471
    IF fl.flhead = 0 THEN                                               03472
      BEGIN %create a header for it%                                    03473
      pgindex ← lodrfb(0, -1); %get a block%                            03474
      frzblk(pgindex, 1);                                               03475
      ON SIGNAL ELSE frzblk(pgindex, -1); %release frozen page%         03476
      corpst[pgindex].ctfile ← fileno;                                  03477
      fl.flhead ← pgindex;                                              03478
      %initialize header of new block%                                  03479
      &fb ← crpgad[pgindex];                                            03480
      %pmap jsys to make the header page zero of the file%              03481
        % r1 ← JFN in left half, 0 in right half %                      03482
        IF fl.flpart THEN                                               03483
          BEGIN                                                         03484
          r1.LH ← fl.flpart;                                            03485
          r3 ← 14B10;                                                   03486
          END                                                           03487
        ELSE                                                            03488
          BEGIN                                                         03489
          IF NOT (r1.LH ← fl.florig) THEN                               03490
            err($"NLS system error");                                   03491
          r3 ← 100488;                                                  03492
          END;                                                          03493
        r1.RH ← 0;                                                      03494
        r2.LH ← 4B5;                                                    03495
        r2.RH ← &fb / 512;                                              03496
        !JSYS pmap;                                                     03497
      fb.fbind ← fb.fbpnum ← 0;                                         03498
      fb.fbtype ← hdtyp;                                                03499
      END;                                                              03500
    filehead[fileno] ← header ← filhdr(fileno);                         03501
    mvbfbf($filhed, header, $filhde-$filhed);                           03502
    %clear core used by file except file header%                        03504
      clrcor(fileno);                                                   03503
    RETURN END.                                                         03505
                                                                        03506
  % build dummy statement %                                             04404
    (gdmys) PROCEDURE (ptr);                                            04405
      LOCAL STRING fname[150], datestr[50];                             04406
      REF ptr;                                                          04407
      *fname* ← NULL;                                                   04408
      IF NOT ptr.stastr THEN filnam(ptr.stfile, $fname);                04409
      *datestr* ← NULL;                                                 04410
      <AUXCOD, getdat>($datestr);                                       04411
      ST ptr ← *fname*, ", ", *datestr*, SP, *initsr*, " ;;;;";         04412
      RETURN;                                                           04413
      END.                                                              04414
                                                                        04415
%.....system file I/O interaction (JFNs, OPEN/CLOSE JSYS STUFF).....%
                                                                        03908
  (sysopen) PROCEDURE % open jsys-- set up registers for and execute
```

```
openf JSYS.  Return TRUE if successful, FALSE if failure with error
message in string whose address was passed as fourth parameter. %
                                                                    0782
    (jfn,  % jfn of file to be opened. %                           03976
    accessmode, % Access desired: read, write, readwrite, or append %
                                                                    03977
    type,  % chrtyp, bintyp, random, comtyp, lpttype %             03978
    astrng % Address of string for error message %                 03979
    );                                                             03980
    LOCAL STRING locstr[60];                                       01796
    LOCAL bp, error, syflag, count, cnttim;                         0783
    REF astrng;                                                    01794
    FOR cnttim←0 UP                                                02606
    UNTIL >1 DO                                                    02607
       BEGIN                                                       02608
       %open file%                                                  0784
       r1 ← jfn;                                                    0785
       CASE accessmode OF                                           0786
          = read: syflag ← 2B5;                                     0787
          = write: syflag ← 1B5;                                    0788
          = readwrite: syflag ← 3B5;                                0789
          = append: syflag ← 2B4 + 1B5;                            03294
          = rwthawed: syflag ← 302B3;                              03837
          = rthawed: syflag ← 202B3;                               07084
          ENDCASE                                                   0790
             BEGIN                                                 02300
             *astrng* ← "NLS system error";                        02301
             RETURN(FALSE);                                        02302
             END;                                                  02303
       CASE type OF                                                 0791
          = chrtyp: r2 ← syflag .V 7B10;                            0792
          = bintyp: r2 ← syflag .V 44B10;                           0793
          = random: r2 ← syflag;                                    0794
          = comtyp: r2 ← syflag .V 7B10;                           01455
          =lpttype: r2 ← syflag .V 7B10;                           02854
          = netype: r2 ← syflag .V 1024B8;                         06706
          ENDCASE                                                   0795
             BEGIN                                                 01928
             *astrng* ← "NLS system error";                        01929
             RETURN(FALSE);                                        01930
             END;                                                  01931
       r2 ← r2 .V 2B2; %Never wait-- error if file busy.%           01715
       IF NOT SKIP !JSYS openf THEN                                  0796
          IF NOT cnttim THEN                                       02610
             BEGIN %dismiss and try again-- TENEX timing problem%
                                                                    02611
             r1 ← 1000;                                             02612
             !JSYS dismS;                                           02613
             END                                                    02615
          ELSE                                                      02616
             BEGIN                                                  0797
             error ← r1;                                            0798
             jfnstr(jfn, $locstr);                                 01795
             CASE error OF                                         01787
                = $opnx1: *astrng* ← *locstr*, " already open";    01788
                = $opnx2:                                          01789
```

```
                    IF accessmode=read THEN                          02913
                       *astrng* ← *locstr*, " does not exist"        02912
                    ELSE                                             02914
                       *astrng* ← "You cannot write on ", *locstr*;
                                                                     02915
                 = $opnx9: *astrng* ← *locstr*, " is busy";          01790
                 = $opnx10: *astrng* ← "File space allocation
                 exceeded";                                          02911
                 ENDCASE *astrng* ← *locstr*, " cannot be opened";
                                                                     01791
              RETURN(FALSE);                                          0809
              END                                                     0810
        ELSE EXIT;                                                   02617
        END;                                                         02609
    %set byte size for write files until tenex bug fixed%            01748
    IF (accessmode = write) AND (type # random) AND (type # lpttype)
    THEN                                                             01749
        BEGIN                                                        01750
        CASE type OF                                                 01751
           =  chrtyp: r3 ← 7B8;                                      01752
           =  bintyp: r3 ← 44B8;                                     01753
           = comtyp: r3 ← 7B8;                                       01755
           = lpttype: r3 ← 7B8;                                      02855
           ENDCASE                                                   01932
              BEGIN                                                  01933
              *astrng* ← "NLS system error";                         01934
              RETURN(FALSE);                                         01935
              END;                                                   01936
        r1.LH ← 11B;  %offset to word to be changed%                 01757
        r1.RH ← jfn;                                                 01758
        r2 ← 77B8;  %bits to be changed%                             01759
        !JSYS chfdb;                                                 01760
        END;                                                         01761
    RETURN(TRUE);                                                     0811
    END.                                                              0813
                                                                      0814
(sysclose) PROCEDURE % Issue close jsys-- Return TRUE if file
closed, FALSE if not with error message in string whose address was
passed.%                                                              0815
    (jfn,  % Of file to be closed %                                  03982
    astrng % Address of string for error message %                   03983
    );                                                               03984
    LOCAL bp, error, count;                                           0816
    LOCAL STRING locstr[60];                                         01807
    REF astrng;                                                      01809
    %close file%                                                      0817
        r1 ← jfn;                                                     0818
        IF NOT SKIP !JSYS closf THEN                                  0819
           BEGIN                                                      0820
           jfnstr(jfn, $locstr);                                     01806
           *astrng* ← *locstr*, "cannot be closed";                  01804
           RETURN(FALSE);                                             0832
           END;                                                       0833
    RETURN(TRUE);                                                     0834
    END.                                                              0836
                                                                      0837
```

```
(lgetjfn)  PROCEDURE   % long gtjfn jsys-- set up registers and
execute long gtjfn.  Returns jfn for file whose name is specified in
the parameters if successful, FALSE with error message in string
whose address is passed if not.  If &dirnam is 0, then the file name
is checked to see if it is a Journal item and the default directory
is taken as the one the user is connected to. %                    0838
   (dirnam,  % string containing default directory -- or 0 %      02775
   flnam,    % string containing the name of the file; may be     02776
      changed by checkjfile if this is a journal item %           04110
   extnsn,   % string containing the default extension %          02777
   flags,    % flags for the gtjfn jsys %                         02778
   errstr);  % string in which to return error messages %         02779
   LOCAL temp,jfilef;                                              0840
   LOCAL STRING lfilnam[200];   % so don't change actual parameters
   %                                                              02783
   REF dirnam, flnam, extnsn, errstr;                              0841
   *lfilnam* ← *flnam*;                                           02785
   IF &dirnam = 0 THEN jfilef ← checkjfile ($lfilnam,flags,FALSE);
                                                                  02027
   (lgetjl):                                                      03199
   gtjfnt[0] ← flags;                                              0843
   gtjfnt[1] ← 377777377777B; %no input or output tty%             0845
   gtjfnt[2] ← 0; %normal device default%                          0852
   gtjfnt[3] ← IF &dirnam = 0 THEN 0 ELSE chbptr (empty) + &dirnam;
                                                                   0853
      %directory default%                                         02786
   gtjfnt[4] ← 0; %normal name default%                            0854
   gtjfnt[5] ← chbptr(empty) + &extnsn;                            0855
   gtjfnt[6] ← 0; %normal protection default%                      0856
   gtjfnt[7] ← 0; %normal account default%                         0857
   gtjfnt[8] ← 0; %not used%                                       0858
   *lfilnam* ← *lfilnam*, 0;                                       0859
   lfilnam.L ← lfilnam.L - 1;                                      0860
   r2 ← chbptr(empty) + $lfilnam;                                  0861
   r1 ← $gtjfnt;                                                   0862
   rubabt ← TRUE;                                                 01539
   IF NOT SKIP !JSYS gtjfn THEN                                    0863
      BEGIN                                                        0864
      temp ← r1;                                                   0865
      rubabt ← FALSE;                                             01540
      IF NOT jfilef AND (jfilef←checkjfile($lfilnam,flags,TRUE))
      THEN GOTO lgetjl;                                           03200
      gtjerr (temp, &errstr, flags);                              01654
      RETURN(FALSE);                                               0875
      END;                                                         0876
   temp ← r1;                                                     02787
   rubabt ← FALSE;                                                01541
   RETURN (temp.RH, temp.LH);                                      0879
   END.                                                            0881
                                                                   0882
(sgtjfn) PROCEDURE % short gtjfn jsys-- set up registers and execute
short gtjfn.  Returns jfn for file whose name is specified in flnam
if successful, FALSE with error message in string whose address is
passed if not.  The file name is checked to see if it is a Journal
item. %                                                            0883
   (flags, % Flags for gtjfn jsys %                               03985
```

```
    flnam,  % String containing name of file; may be changed by   03986
       checkjfile if name is a jcurnal item %                     04109
    errstr  % String for error message %                          03987
    );                                                            03988
    LOCAL errcode, jfn, jfilef;                                    0884
    LOCAL STRING lfilnam[100];                                    02788
    REF flnam, errstr;                                             0885
    %short gtjfn%                                                  0886
    *lfilnam* ← *flnam*;                                          02789
    jfilef←checkjfile ($lfilnam,flags,FALSE);                     02029
    (sgtjfl):                                                     03202
    *lfilnam* ← *lfilnam*, 0;                                      0887
    lfilnam.L ← lfilnam.L - 1;                                     0888
    r2 ← chbptr(empty) + $lfilnam;                                 0889
    r1 ← flags .V sgtjff;                                          0890
    IF NOT SKIP !JSYS gtjfn THEN                                   0891
       BEGIN                                                       0892
       errcode ← r1;                                               0893
       IF NOT jfilef AND (jfilef←checkjfile($lfilnam,flags,TRUE))
       THEN GOTO sgtjfl;                                         03201
       IF &errstr THEN                                           03025
          BEGIN                                                  07122
          gtjerr (errcode, &errstr, flags);                      07121
          *errstr* ← *lfilnam*, ": ", *errstr*;                  07124
          END;                                                   07123
       RETURN(FALSE, errcode);                                   03026
       END;                                                        0905
    jfn ← r1;                                                     02790
    RETURN (jfn.RH, jfn.LH);                                       0906
    END.                                                           0908
                                                                   0909
(reljfn) PROCEDURE % release the jfn %                            02341
    (jfn % to be released %                                       03989
    );                                                            03990
    r1 ← jfn;                                                     02345
    IF NOT SKIP !JSYS rljfn THEN !JFCL 0;                         02344
    RETURN;                                                       02342
    END.                                                          02343
(gtjerr) PROCEDURE % Convert error word from gtjfn into string %
                                                                  04157
    (error, % Code returned by unsuccessful gtjfn %               04158
    string, % Address of string for error message %               04159
    flags   % Passed to gtjfn (used to check access) %            04160
    );                                                            04161
    REF string;                                                   04162
    *string* ← */ CASE error OF                                   04163
       =$gjfx4: $"Illegal character in file name";                04164
       IN ($gjfx5, $gjfx14), = gjfx25:  $"Illegal file name";     04165
       =$gjfx16:  $"No such device";                              04166
       =$gjfx17:  $"No such directory";                           04167
       =$gjfx18:  $"No such file name";                           04168
       =$gjfx19:  $"No such extension";                           04169
       =$gjfx20: $"No such version";                              04170
       =$gjfx21: $"No such file";                                 04171
       =$gjfx22: $"Can't reference any more files";               04172
```

```
      =$gjfx23: $"No room in directory";                                  04173
      =$gjfx24:                                                          04174
         IF flags.LH .A gtjwrt THEN                                      04175
            $"You cannot create a new file in this directory"            04176
         ELSE                                                            04177
            $"File Not On-line;
            If Archived, Use EXEC's INTERROGATE ";                       04178
      ENDCASE $"System file error" )*;                                   04179
   RETURN;                                                               04180
   END.
                                                                         04181
(getgtjflg) PROCEDURE %get flags for gtjfn JSYS%                          0910
   (accessmode,  % Access desired for file %                             03995
   chkpc,  % If chkpcf (=TRUE), this is for a PC: No access by           03996
      other forks.  If orgiff (=FALSE), this is an original file:
      access by other forks permitted. %                                 03999
   version % for right half of flagword (see JSYS manual).               03997
      Oldversion (-4) treated differently.: no longer permitted turn
      off BO, put 0 (highest existing verion ) in RH. %                  04000
   );                                                                    03998
   LOCAL gtflag;                                                          0912
   CASE version OF                                                       03618
      = oldvrsn:  % Old -4 right half; BO must be off, RH 0 now %
                                                                         03619
         BEGIN                                                           03620
         % LH; BO must be off. %                                         03625
         IF chkpc THEN                                                   03626
            gtflag.LH ← gtjcpc % No access by other forks %              03627
         ELSE                                                            03628
            gtflag.LH ←                                                  03629
               (IF accessmode = read THEN gtjred ELSE 0);                03630
         % RH of flag word %                                            03631
            gtflag.RH ← 0; % Get highest existing version because BO
            off. %                                                       03632
         END;                                                            03621
      ENDCASE  % defltvrsn, -1, -2, etc.  %                              03622
         BEGIN                                                           03633
         % LH %                                                          03634
         IF chkpc THEN                                                   03635
            gtflag.LH ←                                                  03636
               gtjcpc .V (IF accessmode = read THEN gtjred ELSE
               gtjigdel)                                                 03643
         ELSE                                                            03637
            gtflag.LH ←                                                  03638
               (IF accessmode = read THEN gtjred ELSE gtjwrt);           03639
         % RH of flag word %                                            03640
            gtflag.RH ← version;                                         03641
         END;                                                            03642
   RETURN(gtflag);                                                        0929
   END.                                                                   0930
                                                                          0931
%.....file status table management routines.....%
                                                                         01120
   (addfil) PROCEDURE % Add a file to file status table. Returns NLS
   file number if successful; calls err if global filcnt is invalid,
   abort if too many files are open. %                                   01136
```

```
   (jfn,  % of file to be added to filst.   If 0, florig field
   remains unchanged (zero?) %                                          04001
   astrng % Address of string containing full file name %               04009
   ):                                                                   04010
   %a new file is added to the file status table.  Astrng               01137
   is the address of a string containing ( FOR NOW) a full               01138
   file name.  If jfn > 0 then it is stored into florig.%               01139
   %-----------------%                                                  01140
   LOCAL fl, unused, fileno, end;                                       01141
   REF fl, astrng;                                                      01142
   IF filcnt NOT IN [0,filmax] THEN err($"NLS system error");           01143
   unused ← FALSE;                                                      01144
   end ← (&fl ← $filst-filstl) + filcnt*filstl;                         01145
   UNTIL (&fl ← &fl + filstl) > end DO                                  01146
      IF NOT fl.flexis THEN                                             01147
         BEGIN                                                          01148
         unused ← TRUE;                                                 01149
         EXIT;                                                          01150
         END;                                                           01151
   IF NOT unused THEN                                                   01152
      BEGIN                                                             01153
      IF filcnt = filmax THEN err($"Too many files open");              01154
      &fl ← (filcnt := fileno ← filcnt + 1)*filstl + $filst;            01155
      END                                                               01156
   ELSE fileno ← (&fl - $filst)/filstl + 1;                             01157
   %zero the filst entry%                                               01158
      clrfil(fileno);                                                   01159
   IF fl.flastr THEN freestring( fl.flastr := 0, $dspblk);              06679
   fl.flastr ← getstring( astrng.L + 1, $dspblk);                       06667
   *[fl.flastr]* ← *astrng*;                                            01160
   IF jfn > 0 THEN fl.florig ← jfn;                                     01161
   fl.flexis ← TRUE;                                                    01167
   IF debugaccess THEN fl.flaccm ← access;                              01633
   %Get Directory number for file%                                      02319
      fldirectory(&fl);                                                 02320
   RETURN (fileno);                                                     01168
   END.                                                                 01169
                                                                        01170
(delfil) PROCEDURE % Delete file (which has been closed) from NLS
internal bookeeping including file status table,  display area
table, and other arrays. Write out called here and in several places
before it (see close)!  Clean this up.  %                               01171
   (fileno % of file to be deleted. %                                   04002
   ):                                                                   04011
   %-----------------%                                                  01175
   LOCAL fl, end, da;                                                   01176
   REF fl, da;                                                          01177
   IF filcnt NOT IN [0, filmax] OR fileno NOT IN [1, filcnt]            01178
   THEN RETURN;                                                         01179
   &fl ← flntadr(fileno);                                               01180
   %must write a routine that will release all frozen dstatements
   from a particular file, in all display windows -- (relall doesn't
   do it)%                                                              01685
   %relall(fileno);% %release any frozen statements%                    01181
   IF NOT writeout(fileno, $lit) THEN err($lit);                        01182
   %clean up window array%                                              01183
```

```
    end ← (&da ← $dpyarea) + dal*dacnt;                                 01184
    UNTIL &da = end DO                                                  01185
       BEGIN                                                            01186
       IF da.daexis AND NOT da.daempty AND                              01187
       da.dacsp NOT= endfil AND                                         01188
       da.dacsp.stfile = fileno THEN                                    01189
          BEGIN                                                         01190
          da.dacsp ← endfil;                                            01191
          da.daempty ←  TRUE;                                           01192
          END;                                                          01193
       &da ← &da + dal;                                                 01194
       END;                                                             01195
    %clean up filst entry%                                              01196
    IF fl.flpart OR fl.florig THEN %error condition -- get rid of
    jfns%                                                               01197
       BEGIN                                                            01198
       IF fl.flpart THEN                                                01695
          BEGIN                                                         01696
          r1 ← fl.flpart;                                               02349
          IF NOT SKIP !JSYS closf THEN reljfn(fl.flpart);               01199
          END;                                                          01700
       IF fl.florig THEN                                                01201
          BEGIN                                                         01701
          r1 ← fl.florig;                                               02346
          IF NOT SKIP !JSYS closf THEN reljfn(fl.florig);               01703
          END;                                                          01706
       END;                                                             01203
    fl.flpcread ← fl.flpart ← fl.florig ← fl.flhead ← 0;                01213
    filepart[fileno] ← FALSE;                                           03315
    fl.flexis ← FALSE;                                                  01215
    IF fl.flastr THEN freestring( fl.flastr := 0, $dspblk);             06668
    IF fl.flpcst THEN freestring( fl.flpcst := 0, $dspblk);             06669
    IF fileno = filcnt THEN BUMP DOWN filcnt;                           01218
    %take care of swork if necessary%                                   02350
       IF NOT swork.stastr AND swork.stfile = fileno THEN               02351
          swork ← endfil;                                               02352
       %similar to <NLS, FILMNP, frerng>.  ensures that fechcl won't
       try to use a bad stid from swork%                                02353
    RETURN;                                                             01219
    END.                                                                01220
                                                                        01221

(cpyfle) PROCEDURE % copy file status table entry for newfno to
oldfno.  Used in outfile to force "new" file to use "old" file
number and filst. %                                                     01277
    (newfno, % file whose status table entry is to be copied %          04003
    oldfno   % file whose filst entry is to be copied to %              04012
    );                                                                  04013
    LOCAL fl1, fl2;                                                     01279
    REF fl1, fl2;                                                       01280
    &fl1 ← flntadr(newfno);                                             01281
    &fl2 ← flntadr(oldfno);                                             01282
    fl2.flexis ← fl1.flexis;                                            01283
    fl2.flhead ← fl1.flhead;                                            01284
    filehead[oldfno] ← filehead[newfno];                                01714
    fl2.flpart ← fl1.flpart;                                            01288
    fl2.flpcread ← fl1.flpcread;                                        03575
```

```
   filepart[oldfno] ← filepart[newfno];                                  03316
   fl2.florig ← fl1.florig;                                              01289
   IF fl2.flastr THEN freestring( fl2.flastr := 0, $dspblk);             06670
   fl2.flastr ← getstring( [fl1.flastr].L, $dspblk);                     06671
   *[fl2.flastr]* ← *[fl1.flastr]*;                                      01292
   IF fl2.flpcst THEN freestring( fl2.flpcst := 0, $dspblk);             06672
   fl2.flpcst ← getstring( [fl1.flpcst].L, $dspblk);                     06673
   *[fl2.flpcst]* ← *[fl1.flpcst]*;                                      01851
   fl2.fldirno ← fl1.fldirno;                                            01290
   RETURN;                                                               01295
   END.                                                                  01296
                                                                         01297
(flntadr) PROCEDURE % Returns the address of the file list entry for
the file number passed %                                                 01318
   (fileno % NLS file number whose filst entry address is sought %
                                                                         04004
   );                                                                    04014
   LOCAL fl; REF fl;                                                     07103
   IF fileno NOT IN [1, filcnt] THEN                                     07076
      err($"NLS system error: illegal file # passed to FLNTADR");
                                                                         07077
   &fl ← (fileno-1)*filstl + $filst;                                     01320
   %IF NOT fl.flexis THEN                                                07105
      err($"NLS system error: illegal file # passed to FLNTADR");%
                                                                         07106
   RETURN(&fl);                                                          07104
   END.                                                                  01321
                                                                         01322
(filnam) PROCEDURE % Get file name from filst entry for file whose
file number is passd.  Put into string whose address is passed %
                                                                         01222
   (fileno, % file number whose name string is sought %                  04005
   astrng   % address of string into which name is placed %              04015
   );                                                                    04016
   %retrieve the file name for the given file number from                01223
   the filest%                                                           01224
   ?/------------------%                                                 01225
   LOCAL fl, fjfn;                                                       01226
   LOCAL STRING locstr[200];                                             06680
   REF fl, astrng;                                                       01227
   &fl ← flntadr(fileno);                                                01228
   IF (fjfn ← fl.florig) = 0 THEN                                        07078
      err($"NLS system error: illegal fst entry detected in
      FILNAM");                                                          07079
   jfnflink( fjfn, $locstr, 011110B6+1 );                                06681
   *astrng* ← *astrng*, *locstr*;                                        01229
   RETURN;                                                               01230
   END.                                                                  01231
                                                                         01232
(filnum) PROCEDURE % Get NLS file number from filst given file name
string.  Returns -1 if error or string not in table. %                   01233
   (astrng % Address of string containingfull file name to               04006
      be compared with names in filst. %                                 04018
   );                                                                    04017
   %------------------%                                                  01236
   LOCAL fl, end;                                                        01237
```

```
   REF fl;                                                          01238
   IF filcnt = 0 OR astrng = -1 THEN RETURN (-1);                   01239
   IF filcnt NOT IN [1,filmax] THEN err($"NLS system error");       01240
   end ← (&fl ← $filst) + filcnt*filstl;                            01241
   DO IF fl.flexis AND                                              01242
      compas(fl.flastr, astrng) THEN                                01243
         RETURN((&fl-$filst)/filstl+1)                              01244
   UNTIL (&fl ← &fl + filstl) >= end;                               01245
   RETURN(-1);                                                      01246
   END.                                                             01247
                                                                    01248
(fldirectory) PROCEDURE % Get directory number given address of
filst entry. %                                                      02286
   (fl % Address of filst entry. %                                  04007
   );                                                               04019
   LOCAL TEXT POINTER ptrl, ptr2;                                   02287
   LOCAL STRING dirs[40];                                           02288
   REF fl;                                                          02308
   FIND SF(*[fl.flastr]*) ['<] ↑ptrl ['>] < CH ↑ptr2;               02269
   *dirs* ← ptrl ptr2;                                              02290
   fl.fldirno ← transdir($dirs);                                    02295
   RETURN;                                                          02310
   END.                                                             02309
(transdir) PROCEDURE % Translates string to directory number using
stdir jsys.  Accepts address of a string, appends 0 to it as
required by TENEX.  If not a match calls err.  Else returns
directory number.%                                                  03208
   (string % Address of directory string; zero will be appended (so
                                                                    04008
      must have room!) %                                            04021
   );                                                               04020
   REF string;                                                      03211
   *string* ← *string*, 0;                                          03210
   r1 ← 1; %exact match%                                            02291
   r2 ← chbptr(empty) + $string;                                    02292
   .JSYS stdir; %stdir%                                             02293
   GOTO dirqm;                                                      02294
   GOTO diram;                                                      03214
   RETURN(r1.RH);                                                   03215
   (dirqm):  err($"No such directory");                             03213
   (diram): err($"Ambiguous directory specification");              03212
   END.                                                             03216
(getenf)    % get EXTERNAL NAME LINK FILE name %                    06707
   PROCEDURE                                                        06708
      (fn,  % file number for which we want the ENLF name %         06709
      astr);         % astr to get the ENLF %                      06710
   LOCAL TEXT POINTER stid, tps, tp1, tp2;                          07066
   LOCAL adstr[40];                                                 07067
   REF astr;                                                        06711
                                                                    06712
   % RETURNS %                                                     07068
      % FALSE: if it cannot find an EXTERNAL NAME LINK FILE name
                                                                    07069
      TRUE: if it finds an EXTERNAL NAME LINK FILE name             07070
```

```
        searches the origin statement of the file whose file number
        it is passed for the string "EXTERNAL LINKS:" followed by
        any number of spaces, tabs, carriage returns, linefeeds,
        and or end-of-lines.  If a link is present here it is
        considered to be the EXTERNAL NAME LINK FILE.  If these
        conditions are not met, then it assumes that the global
        string enlfstr (in user-option page) is the name of the
        EXTERNAL NAME LINK FILE iff enlfstr is NOT null.          07071
    %                                                             07073
                                                                  07074
    stid ← origin;  stid.stfile ← fn;  stid[1] ← 1;               07044
    IF (fn NOT= endfil.stfile) AND                                07047
       FIND SF(stid) ["EXTERNAL LINKS:"] $(SP/TAB/CR/LF/EOL) ↑tps
       THEN                                                       07085
          BEGIN                                                   07048
          ON SIGNAL ELSE                                          07049
             BEGIN                                                07054
             ON SIGNAL ELSE;                                      07055
             GOTO noenlf;                                         07053
             END;                                                 07056
          lnkprs( $tps, $adstr );                                 07050
          ON SIGNAL ELSE;                                         07058
          tp1 ← adstr[ls];  tp1[1] ← adstr[ls+1];                 07051
          tp2 ← adstr[le];  tp2[1] ← adstr[le+1];                 07052
          IF tp1[1] # tps[1] THEN GOTO noenlf;                    07057
          *astr* ← tp1 tp2;                                       07059
          END                                                     07060
    ELSE                                                          07061
      .(noenlf):  BEGIN                                           07062
       IF NOT enlfstr.L THEN RETURN( FALSE );                     07063
       *astr* ← *enlfstr*;                                        07064
       END;                                                       07065
    RETURN( TRUE );                                               06718
                                                                  06719
    END.                                                          06720

%.....status table initialization utility procedures.....%        03975

  (clrfil) PROCEDURE %Clear the filst entry for the given fileno.%
                                                                  01122
    (fileno % file number whose filst entry will be cleared. %    04022
    );                                                            04023
    %----------------%                                           01124
    LOCAL fl;                                                     01125
    REF fl;                                                       01126
    &fl ← flntadr(fileno);                                        01127
    fl.flexis ← FALSE;                                            01128
    fl.flhead ← fl.flpcread ← fl.flpart ← fl.florig ← fl.fllock ← 0;
                                                                  01130
    filepart[fileno] ← FALSE;                                     03314
    filehead[fileno] ← 0;                                         01713
    IF fl.flastr THEN freestring( fl.flastr := 0, $dspblk);       01132
    IF fl.flpcst THEN freestring( fl.flpcst := 0, $dspblk);       06674
    RETURN;                                                       01133
    END.                                                          01134
                                                                  01135
```

```
(clrngs) PROCEDURE % clear ring status table; could be combined with
almost identical cldtbs %                                          01042
   (rn % Address of ring status table in dummy file header %       03970
   );                                                              03971
   LOCAL rngind;                                                   01043
   REF rn;                                                         01044
   rngl ← rngind ← 0;                                              01045
   DO BEGIN                                                        01046
      rn ← 0;                                                      01047
      BUMP &rn;                                                    01048
      END                                                          01049
   UNTIL (rngind ← rngind+1) = rngm;                               01050
   RETURN END.                                                     01051
                                                                   01052
(cldtbs) PROCEDURE % clear data status table; could be combined with
almost identical clrngs %                                          01053
   (dt % Address of data block status table in dummy file header %
                                                                   03973
   );                                                              03974
   LOCAL dtbind;                                                   01054
   REF dt;                                                         01055
   dtbl ← dtbind ← 0;                                              01056
   DO BEGIN                                                        01057
      dt ← 0;                                                      01058
      BUMP &dt;                                                    01059
      END                                                          01060
   UNTIL (dtbind ← dtbind+1) = dtbm;                               01061
   RETURN END.                                                     01062
                                                                   01063
(clrcor) PROCEDURE  % clear core page status table entries and
remove pages from core pertaining to this file (except for the
header, the address of which will be returned.).  Used in
initialization of file.  Similar to writeout! %                    01249
   (fileno % number of file whose corpst entries are to be cleared;
                                                                   04024
      if =-1, the whole corpst is to be cleared. %                 04026
   );                                                              04025
   LOCAL pgindex, ct, hdindex;                                     01252
   REF ct;                                                         01253
   pgindex ← 1;                                                    01254
   hdindex ← -1;                                                   01255
   &ct ← $corpst + 1;                                              01256
   DO BEGIN                                                        01257
      IF ct.ctfull AND                                             01258
         (fileno = -1 OR fileno = ct.ctfile) THEN                  01259
            IF ct.ctpnum = 0 THEN hdindex ← pgindex                01260
            ELSE                                                   01261
               BEGIN                                               01262
               %get rid of this page%                              01263
                  r1 ← -1;                                         01264
                  r2.LH ← 4B5;                                     01265
                  r2.RH ← crpgad(pgindex) / 512;                   01266
                  r3 ← 1B6; % get rid of it %                      01267
                  !JSYS pmap;                                      01268
               ct ← 0;                                             01269
               END;                                                01270
```

```
      BUMP &ct;                                                    01271
      END                                                          01272
   UNTIL (pgindex ← pgindex+1) > rfpmax;                           01273
   RETURN(hdindex);                                                01274
   END.                                                            01275
                                                                   01276
%.....Correspondence list utility procedures....%
                                                                   03861
(unlkclist) PROCEDURE; % after unlock file check clist entries for
validity; clean up if necessary %                                  02496
   LOCAL list, listnd, org;                                        02497
   POINTER list;                                                   02501
   IF clstad = 0 OR /clstad/.clbuff = 0 THEN RETURN;               02498
   list ← /clstad/.clbuff;                                         02499
   listnd ← list + /clstad/.clcnt*cll;                             02502
   DO                                                              02503
      IF NOT <FILMNP, goodrng>(list.clstl) THEN                    02504
         BEGIN                                                     02505
         list.clst2 ← list.clstl;                                  02506
         list.clst2.stpsid ← origin;                               02511
         list.clstl ← endfil;                                      02507
         END                                                       02508
   UNTIL (list ← list + cll) = listnd;                             02509
   RETURN END.
                                                                   02500
(clsid) PROCEDURE %change stid's in passed correspondence list to
sid's. see clhdr and clistr for format of clist.%                   0393
   (list  % Address of corespondence list %                        03862
   );                                                              03863
   %-------------------%                                           0396
   LOCAL cl, end, fnum;                                            0397
   REF cl, list;                                                   0398
   end ← (&cl ← list.clbuff) + list.clcnt*cll;                     0399
   UNTIL &cl = end DO                                              0400
      BEGIN                                                        0401
      IF cl.clstl NOT= endfil THEN                                 0402
         BEGIN                                                     02857
         fnum ← cl.clstl.stfile;                                   0403
         cl.clstl.stsid ← getsid(cl.clstl);                        02861
         cl.clstl.stsidf ← fnum;                                   02909
         END;                                                      02859
      IF cl.clst2 NOT= endfil THEN                                 0404
         BEGIN                                                     02858
         fnum ← cl.clst2.stfile;                                   02862
         cl.clst2.stsid ← getsid(cl.clst2);                        0405
         cl.clst2.stsidf ← fnum;                                   02910
         END;                                                      02860
      &cl ← &cl + cll;                                             0406
      END;                                                         0407
   RETURN;                                                         0408
   END.
                                                                   0409
(clpsid) PROCEDURE % change sid's in passed correspondence list to
stid's.  Uses lookup to find the ring element with the corresponding
sid. see clhdr and clistr for format of clist.%                     0411
   (list % address of correspondence list %                        03864
```

```
    ):                                                              03865
    %-------------------%                                            0414
    LOCAL cl, end;                                                   0415
    LOCAL TEXT POINTER ptr;                                         01779
    REF cl, list;                                                    0416
    ptr ← 0;                                                         0417
    ptr[1] ← 1;                                                      0417'
    ptr.stpsid ← origin;                                             0419
    end ← (&cl ← list.clbuff) + list.clcnt*cll;                      0420
    UNTIL &cl = end DO                                               0421
       BEGIN                                                         0422
       IF cl.clstl NOT= endfil THEN                                  0423
          BEGIN                                                      0424
          ptr.stfile ← cl.clstl.stsidf;                              0425
          lookup($ptr, cl.clstl.stsid, sid);                         0426
          IF (cl.clstl ← ptr) = endfil THEN                          0427
             err($"NLS system error");                              01780
          END;                                                       0428
       IF cl.clst2 NOT= endfil THEN                                  0429
          BEGIN                                                      0430
          ptr.stfile ← cl.clst2.stsidf;                              0431
          lookup($ptr, cl.clst2.stsid, sid);                         0432
          IF (cl.clst2 ← ptr) = endfil THEN                          0433
             err($"NLS system error");                              01781
          END;                                                       0434
       &cl ← &cl + cll;                                              0435
       END;                                                          0436
    RETURN;                                                          0437
    END.                                                             0438
                                                                     0439

(clflup) PROCEDURE % change clist references to oldfil to newfil
(assumes list in sid format) %                                      02953
   (oldfil, % number of file whose references are to be changed %
                                                                    03866
   newfil,  % NLS file number of new file %                         03867
   list     %Address of correspondence list %                       03868
   ):                                                               03869
   %-------------------%                                           02955
   LOCAL cl, end;                                                   02956
   REF cl, list;                                                    02958
   end ← (&cl ← list.clbuff) + list.clcnt*cll;                      02962
   UNTIL &cl = end DO                                               02963
      BEGIN                                                         02964
      IF cl.clstl NOT= endfil THEN                                  02965
         IF cl.clstl.stsidf = oldfil THEN cl.clstl.stsidf ← newfil;
                                                                    02984
      IF cl.clst2 NOT= endfil THEN                                  02972
         IF cl.clst2.stsidf = oldfil THEN cl.clst2.stsidf ← newfil;
                                                                    02985
      &cl ← &cl + cll;                                              02979
      END;                                                          02980
   RETURN;                                                          02981
   END.                                                             02982
                                                                    02983

%.....partial copy and locking utility routines.....%
                                                                    03910
```

```
(lockmes) PROCEDURE % Type out a message indicating that the file is
locked, and by whom %                                                  02829
   (fileno % file number of file whose lock status will be             04035
      printed. %                                                       04062
   );                                                                  04061
   LOCAL fl;                                                           02831
   LOCAL STRING tempsr[150];                                           02841
   REF fl;                                                             02832
   &fl ← flntadr(fileno);                                              02836
   *tempsr* ← "File";                                                  02843
   jgtfdb( fl.florig, 100002AB, $r3);                                  02840
   lockid($tempsr, r3.lkdirn, r3.lkinit);                              02842
   dismes(4000, $tempsr);                                              02844
   RETURN;                                                             02846
   END.
                                                                       02847
(oplock)  PROCEDURE % Open lock lock procedure.  Used with system
routines (such as Journal) to prevent locking conflict by setting
and resetting  the global lock flag. %                                 02428
   (onoff % If TRUE, try to set flag.  If FALSE, try to reset          04036
      flag. %                                                          04064
   );                                                                  04063
   %Waits for open lock lock to be reset, then sets it and returns
   if onoff is true.                                                   02429
      Otherwise, resets lock, and types error message if it was not
      set.                                                             02430
      After waiting 1 minute for lock to be reset, assumes an error
      and overrides  previous lock%                                    02431
   LOCAL failcount;                                                    02432
   IF onoff THEN                                                       02433
      BEGIN                                                            02434
      failcount ← 0;                                                   02435
      LOOP                                                             02436
         BEGIN                                                         02437
         IF flagut(4, $setfg) THEN EXIT; %Got It%                      02440
         IF (failcount ← failcount + 1) > 120 THEN                     02441
            BEGIN %flag has been locked for longer than one minute%
                                                                       02442
            dismes(1, $"Open Lock Flag Overidden--Proceed");           02443
            EXIT LOOP;                                                 02444
            END;                                                       02445
         END;                                                          02448
      END                                                              02449
   ELSE                                                                02450
      IF NOT flagut(4  $rstfg) THEN dismes(1,$"Open Lock Lock Fail
      (Flag Reset)--Proceed");                                         02455
   RETURN;                                                             02457
   END.
                                                                       02458
(setfdb)  PROCEDURE % Set the lock fields in an FDB.  Return TRUE if
access permitted by maywut, FALSE if not%                              02459
   (dirno,% Directory number of person locking file %                  04037
   initls, % Initials of person locking file %                         04065
   fl      % Address of file status table entry for the file. %       04066
   );                                                                  04067
   LOCAL mask, value;                                                  07075
```

```
   REF fl;                                                          02460
   IF NOT maywrt(0, &fl) THEN RETURN(FALSE);                        02461
   mask ← 0;  mask.lkinit ← mask.lkdirn ← 36M;                      02464
   value.lkdirn ← dirno;  value.lkinit ← initls;                    02466
   RETURN( chnfdb( fl.florig, $fdbusw, mask, value) );              02469
   END.
                                                                    02470
(dumpc)  PROCEDURE % Clears or sets all of the pc bits in the file
header ring and data block status tables, depending on value.
Currently, output and update call it after doing all their edits. %
                                                                    03085
   (fileno, % of file whose PC bits are to be set or reset %        04038
   value % Value to be set in PC bits: TRUE or FALSE. (If TRUE,     04066
      then block is to be taken from PC, if FALSE from original
      file. %                                                       04070
   );                                                               04069
   LOCAL fhd, rn, rngblk, dt, stdb, sdbblk;                         03129
   REF rn, dt;                                                      03182
   % get partial copy header address %                             03067
      fhd ← filehead[fileno];                                       03068
   % first do the ring blocks %                                    03136
      &rn ← fhd - $filhed + $rngst;                                 03137
      rngblk ← 0;                                                   03140
      DO                                                            03141
         BEGIN                                                      03142
         IF rn.rfexis THEN rn.rfpart ← value;                       03143
         &rn ← &rn + 1;                                             03156
         END                                                        03157
      UNTIL (rngblk ← rngblk + 1) = rngm;                           03158
   % now do the sdb blocks %                                       03159
      stdb ← 0;                                                     03160
      stdb.strile ← fileno;                                         03161
      &dt ← fhd - $filhed + $atbst;                                 03162
      sdbblk ← 0;                                                   03163
      DO                                                            03164
         BEGIN                                                      03165
         IF dt.rfexis THEN dt.rfpart ← value;                       03166
         &dt ← &dt + 1;                                             03179
         END                                                        03180
      UNTIL (sdbblk ← sdbblk+1) = dtbm;                             03181
   RETURN;                                                          03183
   END.
                                                                    03184
(delpc)  PROCEDURE % delete partial copy of file from user directory
if possible; sets lock fields in fdb to zero (unlocked), set other
flags referring to the existence of the partial copy to zero %  05469
   (fileno, % NLS file number of file whose partial                 05470
      copy is to be deleted %                                       05471
   astrng   % Address of string for error messages %               05472
   );                                                               05473
   %The partial copy is deleted from the user's directory by this
   routine--the header is not reread%                               05474
   %----------------%                                              05475
   LOCAL jfnflg, ckpjfn, dirno, fl;                                 05476
   REF fl, astrng;                                                  05477
   &fl ← flntadr(fileno);                                           05478
```

```
   *astrng* ← NULL;                                                    05479
   IF fileno NOT IN [1,filcnt] OR                                      05480
      filcnt NOT IN [1,filmax] OR                                      05481
      NOT fl.flexis THEN                                               05482
      BEGIN                                                            05483
      *astrng* ← "NLS system error";                                   05484
      RETURN(FALSE);                                                   05485
      END;                                                             05486
   IF NOT fl.flbrws THEN                                               05487
      BEGIN                                                            05488
      IF NOT fl.flpart THEN                                            05489
         BEGIN                                                         05490
         !JSYS gjinf;                                                  05491
         dirno ← r1 := fl.florig;                                      05492
         r2 ← 1000024B; %read one word at 24B%                         05493
         r3 ← 3; %put word into r3%                                    05494
         !JSYS gtfdb;                                                  05495
         END;                                                          05496
      IF fl.flpart OR (r3.lkinit = cinit AND r3.lkdirn = dirno) THEN
                                                                       05497
         IF NOT setfdb(0, 0, &fl) THEN                                 05498
            BEGIN                                                      05499
            *astrng* ← "No write access";                              05500
            RETURN(FALSE);                                             05501
            END;                                                       05502
      END;                                                             05503
   IF fl.flpart THEN                                                   05504
      BEGIN                                                            05505
      %delete partial copy file%                                       05506
      r1 ← fl.flpart .V 4B11;                                          05507
      IF NOT SKIP !JSYS closf THEN                                     05508
         *astrng* ← "Cannot close partial copy";                       05509
      r1 ← fl.flpart;                                                  05510
      IF NOT SKIP !JSYS delf THEN                                      05511
         *astrng* ← "Cannot delete partial copy";                      05512
      END;                                                             05513
   fl.flpart ← 0;                                                      05514
   filepart[fileno] ← FALSE;                                           05515
   RETURN(TRUE);                                                       05516
   END.
                                                                       05517
(unlkfile) PROCEDURE % Unlock file execution procedure %               05436
   (fileno % NLS file number of file to be unlocked %                  05437
   );                                                                  05438
   LOCAL dirno, fl, lkword;                                            05439
   LOCAL STRING msgstr[100];                                           05440
   REF fl;                                                             05441
   &fl ← flntadr(fileno);                                              05442
   !JSYS gjinf; %get current directory%                                05443
   dirno ← r1 := fl.florig;                                            05444
   r2 ← 1000024B; %read word at 24B in fdb%                            05445
   r3 ← 3; %and stuff it in r3%                                        05446
   !JSYS gtfdb;                                                        05447
   lkword ← r3; %lock information%                                     05448
   IF lkword.lkdirn # 0 AND lkword.lkinit # 0 THEN                     05449
      BEGIN                                                            05450
```

```
      IF lkword.lkdirn # dirno OR lkword.lkinit # cinit THEN      05451
         err($"You do not have this file locked.");               05452
      END                                                         05453
   ELSE err($"This file is not locked");                          05454
   ckdiracc(fileno);  %write access, this directory?%             05455
   IF NOT writeout(fileno, $lit)
   OR NOT delpc#[fileno,  lit) THEN err($lit);                     05456
   IF lit.L > empty THEN                                          05457
      BEGIN                                                       05458
      dismes(2, $lit);                                            05459
      END;                                                        05461
   IF NOT rdhdr(fileno, $lit) THEN err($lit);                     05462
   IF lit.L > empty THEN                                          05463
      BEGIN                                                       05464
      dismes(2, $lit);                                            05465
      END;                                                        05467
   RETURN END.
                                                                  05468
(cpcnam)  PROCEDURE % converts the file name passed in oldstg to a
file name for pc in newstng %                                      0958
   (oldstg, % Address of original file name string. %             04039
   dirnum   % Directory number of user creating PC; name will     04072
      go into PC name. %                                          04075
   );                                                             04073
   REF oldstg;                                                     0961
   LOCAL byt, retadr;                                             01731
   LOCAL STRING dirnam[25], newstg[200];                          01725
                                                                  06659
   % RETURNS %                                                   06660
      % returns the address of a string  (which will contain the pc
      name) which is obtained via getstring from block dspblk %   06661
                                                                  06662
   CCPOS SF(*oldstg*);                                             0962
   IF NOT FIND ['<'] tp1 ['>'] tp2 ←p2 tp3 ['.'] tp4 ←p4 [';']     0963
   tp5 ←p5                                                         0964
      THEN err($"NLS system error");                               0965
   %get directory name%                                           01726
      r1 ← byt← cnbptr(empty) + $dirnam;                          01727
      r2 ← dirnum;                                                01728
      IF NOT SKIP !JSYS dirst THEN err($"NLS system error");       01729
      dirnam.L ← slngth(byt, r1);                                 01730
   *newstg* ←'<, *dirnam*, '>,  '(, p1 p2, '),  p3 p4, '.;         0966
   *newstg* ← *newstg*, "PC";                                      0968
   *newstg* ← *newstg*, p5 SE(*oldstg*);                           0972
   retadr ← getstring( newstg.L + 1, $dspblk);                    06663
   *[retadr]* ← *newstg*;                                         06664
   RETURN( retadr );                                              06665
   END.                                                            0974
                                                                   0975
%.....access utility routines.....%
                                                                  03911
(ckdiracc)   PROCEDURE % given a file number, this routine checks
that the current job has write access to the directory that the file
lives in; if not, this routine goes to err with a nasty message %
                                                                  03003
   (fileno % of file to which access is being sought %            04040
```

```
    );                                                              04076
    LOCAL cnt, byt, fl;                                             03005
    LOCAL STRING msgstr[100];                                       03006
    REF fl;                                                         03007
    &fl ← flntadr(fileno);                                          03010
    !chkac( fl.fldirno );                                           04125
    IF (r1 .A 20B) THEN RETURN;                                     04126
    *msgstr* ← "No write access to ";                               03012
    r1 ← byt ← chbptr(msgstr.L) + $msgstr;                          03011
    r2 ← fl.fldirno;                                                03013
    IF SKIP !JSYS dirst AND                                         03014
       (cnt ← slngth(byt, r1)) + msgstr.L <= msgstr.M THEN          03015
          msgstr.L ← msgstr.L + cnt;                                03016
    err($msgstr);                                                   03017
    END.
                                                                    03019
(setaccess)  PROCEDURE % Set the NLS system access for the file
indicated by fileno to type if access type s LEGAL.  Calls err if
anything wrong. %                                                   01634
    (fileno, % file number whose access is to be set. %             04041
    type % Access type desired. %                                   04077
    );                                                              04078
    LOCAL jfn, wrtdnm, wrtini;                                      01636
    IF type > 7 THEN err($"Illegal Access type");                   01653
    %First check for lock and access%                               01637
    r1 ← jfn ← [flntadr(fileno)].florig;                            01638
    r2 ← 100002kB; r3 ← $r3;                                        01639
    !JSYS gtfdb;                                                    01640
    IF NOT access .A accmask[r3.acctyp] THEN                        01641
       err($"Illegal Access");                                      01845
    IF (r3.lkinit # 0) OR (r3.lkdirn # 0) THEN                      01642
       BEGIN %locked..check by whom%                                02304
       wrtdnm ← r3.lkdirn; wrtini ← r3.lkinit;                      01643
       !JSYS gjinf;                                                 01644
       IF r1 # wrtdnm OR cinit # wrtini THEN                        01645
          err($"File Locked");                                      01844
       END;                                                         01672
    IF NOT maywrt(fileno) THEN                                      02305
       err($"No write access");                                     02306
    %Now it is free to change access%                               01646
       r1.LH ← 24B; r1.RH ← jfn;                                    01647
       r2 ← 0;                                                      01651
       r2.acctyp ← 36M;                                             01648
       r3.acctyp ← type;                                            01649
       !JSYS chfdb;                                                 01650
    RETURN END.
                                                                    01652
(enablaccess) PROCEDURE % Set NLS system access  to file or global
mask for user.  (Used for special subsystems , e.g., Journal.) Set
bit in access mask which corresponds to type.  Set mask in file if
fileno # 0, otherwise set global mask. %                            01654
    (fileno, % File number whose access is to be set; if zero       04042
       set global access mask. %                                    04081
    type  % NLS access type %                                       04079
    );                                                              04080
    LOCAL fl; REF fl;                                               01662
```

```
   IF type > 7 THEN err($"Illegal Access Type");                    01657
   IF fileno THEN BEGIN &fl ← flntadr(fileno);                      01658
      fl.flaccm ← fl.flaccm .V accmask[type];                       01659
   END ELSE access ← access .V accmask[type];                       01660
   RETURN END.
                                                                    01661
(disabaccess) PROCEDURE % see enablaccess for comment %             01663
   (fileno, % File number whose access is to be disabled; if zero,
                                                                    04082
      set global access mask. %                                     04083
   type  % NLS access type %                                        04084
   ):                                                               04085
   LOCAL fl, mask; REF fl;                                          01665
   IF type > 7 THEN err($"Illegal Access Type");                    01671
   mask ← -(accmask[type] +1); %one complement%                     01666
   IF fileno THEN BEGIN &fl ← flntadr(fileno);                      01667
      fl.flaccm ← fl.flaccm .A mask;                                01668
   END ELSE access ← access .A mask;                                01669
   RETURN END.
                                                                    01670
(maywrt)      PROCEDURE % Uses chkac JSYS to see if user has access
to the file whose file status table entry address or file number is
passed. %                                                           01961
   (fileno, % file number of file to which access is to be checked.
                                                                    04044
      If 0, assumes file status table address has been passed. %
                                                                    04088
   fl % Address of filst entry for this file; if 0, the address    04086
      will be computer from the fileno? %                           04089
   ):                                                               04087
   %returns TRUE if may write on the specified file %               01965
   REF fl;                                                          01988
   IF fileno THEN &fl ← flntadr(fileno);                            01987
   !chkac( fl.fldirno ); %directory for that file%                  01963
   IF NOT (r1 .A 20B) THEN RETURN (FALSE);                          04127
   %get current user settable word%                                 04129
      !gtfdb( fl.florig, 1000024B, $r3);                            04128
   %return value from attemp to change sign bit in USW in FDB%      04136
      RETURN( chnfdb( fl.florig, 24B, 4B11, r3) );                  01964
   END.
                                                                    03002
(chnfdb)      PROCEDURE % Return true if file fdb changed properly%
                                                                    04130
   (jfn,    %jfn for file%                                          04131
   offset,  %offset into fdb for word to be changed%                04132
   mask,    %mask for which bits to change%                         04133
   value    %new value for word (masked by mask)%                   04134
   ):                                                               04135
   LOCAL savedispatch;                                              04137
   %save the old illegal instruction psi dispatch address%          04138
      savedispatch ← ilsdsp;                                        04139
   %set up new dispatch address%                                    04140
      ilsdsp ← $chndsp;                                             04141
   %try to change the fdb%                                          04142
      r1.LH ← offset;   r1.RH ← jfn;                                04143
      !chfdb( r1, mask, value);                                     04144
```

```
      %successful, now cleanup and return TRUE%                          04145
         ilsdsp ← savedispatch;                                          04146
         RETURN (TRUE);                                                  04147
      %not successful, debrk and return FALSE%                           04148
         (chndsp):                                                       04149
            ilsdsp ← savedispatch;                                       04150
            /levtab/ ← $chnext;                                          04151
            !debrk();                                                    04152
         (chnext):                                                       04153
            RETURN( FALSE );                                             04154
      END.                                                               04155
%.....privacy routines.....%                                             06865
   (chprvsts) %change file's privacy status%
   PROCEDURE (fileno, newprvsts);                                        06866
      %----------%                                                       06867
      LOCAL setting, stid;                                               06868
      LOCAL TEXT POINTER tp1, tp2;                                       06869
      %----------%                                                       06870
      CASE newprvsts OF                                                  06871
         = $pspublic: setting ← FALSE;                                   06872
         = $psprivate: setting ← TRUE;                                   06873
         = FALSE: RETURN;                                                06874
         ENDCASE err (0);                                                06875
      %force creation of partial copy%                                   06876
         stid ← 0;                                                       06877
         stid.stpsid ← origin;                                           06878
         stid.stfile ← fileno;                                           06879
         FIND SF(stid) ↑tp1 CH ↑tp2;                                     06880
         ST tp1  TP2 ← tp1 tp2;                                          06881
      /filehead /fileno/ + $filhde-$filhed/.prvsts ← setting;            06882
      RETURN;                                                            06883
      END.                                                               06884
                                                                         06885

   (rdprvsts) %read file's privacy status%
   PROCEDURE (fileno);                                                   06908
      %----------%                                                       06909
      LOCAL curprvsts;                                                   06910
      %----------%                                                       06911
      CASE /filehead /fileno/ + $filhde-$filhed/.prvsts OF               06912
         = FALSE: curprvsts ← $pspublic;                                 06913
         = TRUE: curprvsts ← $psprivate;                                 06914
         ENDCASE;                                                        06915
      RETURN (curprvsts);                                                06916
      END.                                                               06917
                                                                         06918

   (vrprvacc) %verify ident's privacy-status access to file%
   PROCEDURE (fileno, %or% filename, %or% catstid, ident);               06919
      %----------%                                                       06920
      LOCAL stid, filestid, outcome, fl, derivedfileno;                  06921
      LOCAL STRING docnumber /9/;                                        06922
      LOCAL TEXT POINTER tp1, tp2;                                       06923
      REF filename, ident, fl;                                           06924
      %----------%                                                       06925
      %access check disabled, enabled wheel, or system?%                 06926
         IF skpachk OR enwheel () OR libflg THEN RETURN (TRUE);          06927
      %initialize%                                                       06928
```

```
      stid ← derivedfileno ← 0;                                          06929
      ON SIGNAL ELSE                                                     06930
         BEGIN                                                           06931
         IF stid.stfile AND NOT catstid THEN                             06932
            sysclose (stid.stfile := 0);                                 06933
         IF derivedfileno AND NOT fileno THEN                            06934
            close (derivedfileno := 0);                                  06935
         END;                                                            06936
   %filename supplied%                                                   06937
      derivedfileno ← (IF &filename THEN                                 06938
         open (0, &filename) ELSE fileno);                               06939
   %fileno supplied%                                                     06940
      outcome ← TRUE;                                                    06941
      IF NOT (stid ← catstid) THEN                                       06942
         BEGIN                                                           06943
         %public file?%                                                  06944
            IF rdprvsts (derivedfileno) = $pspublic THEN                 06945
               GOTO vrpexit;                                             06946
         %does origin statement give him access?%                        06947
            %build stid for origin statement%                            06948
               filestid.stpsid ← origin;                                 06949
               filestid.stfile ← derivedfileno;                          06950
            IF NOT getfacc (filestid, 0, $tp1, $tp2)                     06951
            OR ckilmem (&ident, $tp1, $tp2, 0) THEN                      06952
               GOTO vrpexit;                                             06953
         %does catalog entry give him access?%                           06954
            &fl ← fintadr (derivedfileno);                               06955
            IF NOT cnvfndocnum (fl.flastr, $docnumber)                   06956
            OR NOT (stid ← gtcatent ($docnumber, 0, 0, FALSE)) THEN
                                                                         06957
               BEGIN                                                     06958
               outcome ← FALSE;                                          06959
               GOTO vrpexit;                                             06960
               END;                                                      06961
         END;                                                            06962
   %open catalog file supplied%                                          06963
      outcome ←                                                          06964
         IF getcacc (stid, 0, $tp1, $tp2) THEN                           06965
            ckilmem (&ident, $tp1, $tp2, 0)                              06966
         ELSE TRUE;                                                      06967
   %terminate%                                                           06968
      (vrpexit): IF stid.stfile AND NOT catstid THEN                     06969
         close (stid.stfile := 0);                                       06970
      IF derivedfileno AND NOT fileno THEN                               06971
         close (derivedfileno := 0);                                     06972
   RETURN (outcome);                                                     06973
   END.                                                                  06974
                                                                         06975
(getfacc) %fetch access field from file origin statement%
PROCEDURE (originstid, fldstr, ptr1, ptr2);                              06976
   %----------%                                                          06977
   RETURN (getcacc (originstid, fldstr, ptr1, ptr2));                    06978
   END.                                                                  06979
                                                                         06980
(setfacc) %set access field in file origin statement%
PROCEDURE (originstid, repstr, ptr1, ptr2);                              06981
```

```
    %----------%                                                    06982
    RETURN (setcacc (originstid, repstr, ptr1, ptr2));             06983
    END.                                                           06984
                                                                   06985
%.....miscellaneous utility routines.....%
                                                                   03909
(writeout) PROCEDURE % write out all pages of a file and PO; all
pages associated with the file are removed from the user's
map--written back onto the file.  Returns TRUE is successful, FALSE
if error with message in string whose address is passed.  Similar to
clrcor.  collect common code. %                                     0531
   (fileno,  % of file to be written out %                         04027
   astrng  % Address of string for error message %                 04045
   );                                                              04046
   %----------------%                                               0536
   LOCAL pgindex, ct, fl;                                          0537
   REF ct, fl, astrng;                                             0538
   IF fileno NOT IN [1,filcnt] OR filcnt NOT IN [1,filmax]         0539
   THEN                                                            0540
      BEGIN                                                        02000
      *astrng* ← "NLS system error";                               0541
      RETURN(FALSE);                                               02001
      END;                                                         02002
   pgindex ← 1;                                                    0542
   &ct ← $corpst + 1;                                              0543
   DO                                                              0544
      BEGIN                                                        0545
      IF ct.ctfull AND ct.ctfile = fileno THEN                     0546
         BEGIN                                                     0547
         IF ct.ctpnum = 0 AND ct.ctfroz > 1
         OR ct.ctpnum # 0 AND ct.ctfroz > 0 THEN                   0548
            BEGIN                                                  02004
            *astrng* ← "NLS system error";                         02005
            RETURN(FALSE);                                         02006
            END;                                                   02007
         %JSYS to get rid of the page%                             0556
         r1 ← -1;                                                  0557
         r2.LH ← 4B5;                                              0558
         r2.RH ← crpgad[pgindex] / 512;                            0559
         !JSYS pmap;                                               0560
         ct ← 0;                                                   0561
         END;                                                      0562
      BUMP &ct;                                                    0563
      END                                                          0564
   UNTIL (pgindex ← pgindex+1) > rfpmax;                           0565
   &fl ← flntadr(fileno);                                          0566
   fl.flhead ← filehead[fileno] ← 0;                               0567
   RETURN(TRUE);                                                   0568
   END.                                                            0569
                                                                   0570
(rdhdr)    PROCEDURE  % read file header %                         03341
   (fileno, % NLS file number of file whose header is to be        04028
      read into core %                                             04049
   astrng   % Address of string to be used for error message. %    04047
   );                                                              04048
   % To read the header block (block zero) of a file into core, call
```

```
                                                                    03342
this routine with the file number.  If a partial copy               03343
exists for this file, the header is read from it,                   03344
otherwise the header is read from the original file.                03345
The file header block is frozen into core and its address           03346
stored into the filst entry for this file.  If the header           03347
is badand it is from a PC, the file is unlocked and the header
read from the original file.  If there was no PC or if the PC
header was bad and the original file header is bad, the file is
initialized by intfil.  A message is returned in the passed
astring, if something went wrong.  Otherwise the string is set to
null.  Returns TRUE if some header read in (even from an
initialized file), FALSE if system error.  Could also call err
with bfile ("bad file") error number is  the jfn in the file
status table for this file is 0. %                                  03348
%-------------------%                                              03349
LOCAL errnum,  fl, pgindx, vwd, header, blk, db, rn, end, jfn;
                                                                    03350
LOCAL tmp;                                                          07125
REF astrng, fl, vwd, db, rn;                                        03351
&fl ← flntadr(fileno);                                              03352
IF &astrng = 0 THEN &astrng ← $lit;                                 03353
*astrng* ← NULL;                                                    03354
IF fileno NOT IN [1,filcnt] OR                                      03355
   filcnt NOT IN [1,filmax] OR                                      03356
   NOT fl.flexis THEN                                               03357
   BEGIN                                                            03358
   *astrng* ← "NLS system error";                                   03359
   RETURN(FALSE);                                                   03360
   END;                                                             03361
% get a page %                                                      03362
   pgindx ← lodrfb (0, -1);                                         03363
   corpst[pgindx].ctfile ← fileno;                                  03364
   frzblk (pgindx, 1);                                              03365
   ON SIGNAL ELSE frzblk(pgindx, -1); %release frozen page%         03366
% PMAP the header %                                                 03367
   IF jfn ← fl.flpart THEN %get it from the PC%                     03368
      r3 ← ( IF NOT fl.flpcread THEN 14B10 %rw access%  ELSE
      100488 %read access only% )                                   03369
   ELSE                                                             03370
      BEGIN %get it from the original file%                         03371
      IF NOT jfn ← fl.florig THEN err(bfile);                       03372
      r3 ← 100486; %private copy on write%                          03373
      END;                                                          03374
   r1.LH ← jfn;                                                     03375
   r1.RH ← 0;                                                       03376
   blk ← crpgad[pgindx];                                            03377
   r2.RH ← blk / 512;                                               03378
   r2.LH ← 485;                                                     03379
   % r3 already set up %                                            03380
   !JSYS pmap;                                                      03381
fl.flhead ← pgindx;                                                 03382
&vwd ← (header ← filhdr(fileno)) - $filhed + $nlsvwd;               03383
filehead[fileno] ← header;                                          03384
errnum ← 0;                                                         03385
IF [blk].fbtype NOT= hdtyp OR                                       03386
```

```
[blk].fbpnum NOT= 0 OR                                                  03387
vwd # nlsvwd THEN                                                       03388
   IF fl.flpart THEN                                                    03389
      BEGIN                                                             03390
      errnum ← 1;                                                       03391
      IF NOT writeout(fileno, &astrng)
      OR NOT delpc(fileno, &astrng) THEN                                03392
         RETURN(FALSE);                                                 03393
      ON SIGNAL ELSE; %disarm%                                          03394
      %read header from original file%                                  03395
      %get a new page%                                                  03396
      pgindx ← lodrfb(0, -1);                                           03397
      crpst[pgindx].ctfile ← fileno;                                    03398
      frzblk(pgindx, 1);                                                03399
      ON SIGNAL ELSE frzblk(pgindx, -1); %release frozen page%
                                                                        03400
      IF NOT r1.LH ← fl.florig THEN                                     03401
         BEGIN                                                          03402
         *astrng* ← "NLS System Error";                                 03403
         RETURN(FALSE);                                                 03404
         END;                                                           03405
      r1.RH ← 0;                                                        03406
      r3 ← 1004B8; %private copy on write%                              03407
      blk ← crpgad[pgindx];                                             03408
      r2.RH ← blk/512;                                                  03409
      r2.LH ← 4B5;                                                      03410
      !JSYS pmap;                                                       03411
      fl.fihead ← pgindx;                                               03412
      &vwd ← (header ← filhdr(fileno)) - $filhed + $nlsvwd;             03413
      filehead[fileno] ← header;                                        03414
      IF [blk].fbtype # hdtyp OR
      [blk].fbpnum # 0 OR
      vwd # nlsvwd THEN                                                 03415
         BEGIN                                                          03416
         errnum ← 2;                                                    03417
         intfil(fileno);                                                03418
         END;                                                           03419
      END                                                               03420
   ELSE                                                                 03421
      BEGIN                                                             03422
      !gtfdb( fl.florig, 1B6+$fdbbyv, $tmp);                            07126
      IF tmp.RH = 0 OR (tmp.RH = 1 AND iszeropage(blk)) THEN            07127
         BEGIN                                                          07128
         errnum ← 2;                                                    03423
         intfil(fileno);                                                03424
         END                                                            07129
      ELSE                                                              07130
         BEGIN                                                          07131
         *astrng* ← *[fl.flastr]*, " is not an NLS file";               07132
         RETURN( FALSE );                                               07133
         END;                                                           07134
      END;                                                              03425
[header + $rfbs + 1 -$filhed] ← 0; %ca bits%                            03426
end ← (&rn ← $rngst - $filhed + header) + rngm;                         03427
DO                                                                      03428
   BEGIN                                                                03429
```

```
      rn.rfcore ← FALSE;                                                03430
      IF NOT fl.flpart THEN rn.rfpart ← FALSE;                          03431
      END                                                               03432
   UNTIL (&rn ← &rn + 1) >= end;                                        03433
   end ← (&db ← $dtbst - $filhed + header) + dtbm;                      03434
   DO                                                                   03435
      BEGIN                                                             03436
      db.rfcore ← FALSE;                                                03437
      IF NOT fl.flpart THEN db.rfpart ← FALSE;                          03438
      END                                                               03439
   UNTIL (&db ← &db + 1) >= end;                                        03440
   CASE errnum OF                                                       03441
      = 1: *astrng* ← "PC bad, file unlocked";                          03442
      = 2: *astrng* ← NULL; %used to be file initialized%               03443
      ENDCASE;                                                          03444
   RETURN(TRUE);                                                        03445
   END.                                                                 03446
                                                                        03447
(iszeropage) PROCEDURE (page);  %return true if page pointed to by
PAGE is all zeroes, FALSE if not.  note: dos not check for valid
page boundary.%                                                         07135
   REF page;                                                            07136
   LOCAL i;                                                             07137
   FOR i ← 0 UP UNTIL > 511 DO                                          07138
      IF page[i] THEN RETURN(FALSE);                                    07139
   RETURN(TRUE);                                                        07140
   END.                                                                 07141
                                                                        07142
(delovsrns) PROCEDURE % Deletes old versions of file if there are
more than the number passed as the second parameter %                   02539
   (jfn,  % of file whose old versions are to be deleted %              04029
   number % of versions to keep %                                       04050
   );                                                                   04051
   r1 ← jfn;                                                            01774
   r2 ← number;                                                         01775
   IF NOT SKIP !JSYS delnf THEN                                         01776
      BEGIN                                                             02536
      dismes(2, $"unable to delete old versions");                      01777
      END;                                                              02538
   RETURN END.                                                          02541
                                                                        02541
(getversn) PROCEDURE % returns the version number associated with
the file name passed it %                                               0932
   (string % Address of string containing file name %                   04034
   );                                                                   04052
   REF string;                                                          0934
   CCPOS SF(*string*);                                                  0935
   *vrsnno* ← NULL;                                                     0936
   IF FIND (';) ↑p1 THEN                                                0937
      *vrsnno* ← p1 SE(p1);                                             0938
   IF vrsnno.L = empty THEN RETURN(FALSE);                              0939
   RETURN( cvsno(@vrsnno) );                                            0940
   END.                                                                 0941
                                                                        0942
(jfnstr)   PROCEDURE % puts full file name for jfn passed into
string %                                                                0943
```

```
   (jfn, % of file whose name is to be placed into a string %    02806
   string % Address of string into which name will be placed. % 04053
   );                                                           04054
   REF string;                                                   0946
   RETURN (jfntostr (jfn, &string, jfnstf) );                    0955
   END.                                                          0956
                                                                 0957
(jfntostr) PROCEDURE % converts jfn to a full file name %       02792
   (jfn,     % the jfn %                                        02808
   string,   % string into which to put the file name %        02809
   format);  % format of file name to be put out by jfns JSYS % 02810
   LOCAL bytptr, count, temp;                                   02793
   LOCAL STRING lstr[100];                                      02811
   REF string;                                                  02794
   r1 ← bytptr ← chbptr(empty) + $lstr;                         02795
   r2 ← jfn;                                                    02796
   r3 ← format;                                                 02797
   !JSYS jfns;                                                  02798
   temp ← r1;                                                   02812
   count ← slngth (bytptr, temp);                               02799
   IF string.M < count THEN err ($"NLS system error");          02800
   lstr.L ← count;                                              02801
   *string* ← *lstr*;                                           02802
   RETURN;                                                      02803
   END.                                                         02804
                                                                02805
(jfnflink)  % converts jfn to file link string %                06682
   PROCEDURE                                                    06683
      (fjfn,          % jfn of file %                           06684
      jfnname,        % address of string to receive file link % 06685
      flags           % flags for jfns jsys for which fields to put in
      link %                                                    06686
      );                                                        06687
   LOCAL TEXT POINTER                                           06688
      tp1, tp2, tp3, tp4;                                       06689
   LOCAL STRING                                                 06690
      locstr[200]    % work string %                            06691
      ;                                                         06692
   REF jfnname;                                                 06693
                                                                06694
   % fields from jfntostr (+ dir & file names & ;T and punctuation)
   %                                                            06695
      jfntostr( fjfn, $locstr, flags .V 011B9 .V 4B4 .V 1);     06696
   % now parse this string %                                   06697
      IF NOT FIND SF(*locstr*) > /'</ ↑tp1 /'>/ ↑tp2 ←tp2 ↑tp3
      /ENDCHR/ ↑tp4 THEN err( $"system screwup");               06698
   % now build the file link %                                 06699
      *jfnname* ← '<, SP, tp1 tp2, ',, SP, tp3 tp4, ',, SP, '>; 06700
   % all done so go home %                                     06701
      RETURN;                                                   06702
                                                                06703
   END.                                                         06704
(incrvrsn) PROCEDURE (string);                                  04361
   %Given the addreaa of a string containing a file name,  THIS
   ROUTINE WILL INCREMENT THE NUMBER IN THE STRING REPRESENTING THE
```

```
   version  number  (if the string has no ";nn" in it, the routine
   returns FALSE.%                                                  04362
   %------------%                                                  04363
   LOCAL TEXT POINTER tptr;                                         04364
   LOCAL number, jfn, flgbits;                                      04365
   REF string;                                                      04366
   CCPOS SF(*string*);                                              04367
   IF FIND [';] ↑↑tptr THEN                                         04368
      BEGIN                                                         04369
      *string* ← SF(*string*) tptr;                                 04370
      %get new file name%                                           04371
         flgbits ← getgtjflg(write, origff, dfltvrs);               04372
         IF NOT jfn ← sgtjfn(flgbits, &string, $lit) THEN           04373
            BEGIN                                                   04374
            *string* ← "*****";                                     04375
            RETURN(FALSE);                                          04376
            END;                                                    04377
         jfnstr(jfn, &string);                                      04378
         reljfn (jfn);                                              04379
      RETURN(TRUE);                                                 04380
      END                                                           04381
   ELSE                                                             04382
      BEGIN                                                         04383
      *string* ← "*****";                                           04384
      RETURN(FALSE);                                                04385
      END                                                           04386
   END.
                                                                    04387
(gdname)   PROCEDURE % Convert directory number and into a directory
string.  Calls err if error occurs. %                               02322
   (dirno, % TENEX directory number %                               04030
   dname   % Address of string into which directory name is to      04055
      be placed %                                                   04057
   );                                                               04056
   LOCAL length, bp;                                                02327
   REF dname;                                                       02326
   bp ← r1 ← cbptr(empty) + &dname;                                 01733
   r2 ← dirno;                                                      01734
   IF NOT SKIP !JSYS dirst THEN err($"System error");               01735
   length ← slngth(bp, r1);                                         01736
   IF length > dname.M THEN err($"NLS System error");               01737
   dname.L ← length;                                                01738
   RETURN END.
                                                                    02325
(gdftdir)  PROCEDURE % Given a filenumber, puts owner name in a
string %                                                            05518
   (fileno, % File number whose owner name is sought %              05519
   string  % Address of string into which owner name is to be       05520
      placed %                                                      05521
   );                                                               05522
   %changed to handle a pointer in funo (with sign bit) to string in
   file header block instead of file number in funo. 7-DEC-73 23:09
   JDH%                                                             05523
   LOCAL unum, bytptr, unstr, filhaddr;                             05524
   REF string, unstr;                                               05525
   unum ← /($funo - $filhed) + (filhaddr ← filhdr(fileno))/;        05526
```

```
   IF unum < 0                                                          05527
      THEN BEGIN                                                        05528
         filhaddr ← filhaddr .A 777000B;                                05529
         &unstr ← filhaddr + unum.RH;                                   05530
         *string* ← *unstr*;                                            05531
         END                                                            05532
      ELSE BEGIN                                                        05533
         r1 ← bytptr ← chbptr(empty) + &string;                         05534
         r2 ← unum;                                                     05535
         IF NOT SKIP !JSYS dirst THEN                                   05536
            err($"Illegal user number in file header.");                05537
         IF (string.L ← slngth(bytptr, r1)) > string.M THEN             05538
            err($"User name too long.");                                05539
         END;                                                           05540
   RETURN;                                                              05541
   END.
                                                                        05542
(checkjfile) PROCEDURE %convert filename to absolute form via
catalogs%                                                               06986
   (filename, flags, thorough);                                         06987
   %----------%                                                         06988
   LOCAL tempjfn, catstid;                                              06989
   LOCAL STRING docnumber [9], tempfn [40];                             06990
   REF filename;                                                        06991
   %----------%                                                         06992
   %skip catalog search unless read access requested%                   06993
      IF NOT flags.LH .A gtjred THEN RETURN (FALSE);                    06994
   %if superficial search, skip catalogs if directory specified%
                                                                        06995
      IF NOT thorough AND *filename* [1] = '< THEN RETURN (FALSE);
                                                                        06996
   %syntactically a journal document?%                                  06997
      IF NOT cnvfndocnum (&filename, $docnumber) THEN                   06998
         RETURN (NOT thorough);                                         06999
   %initialize%                                                         07000
      catstid ← 0;                                                      07001
      ON SIGNAL ELSE IF catstid.stfile THEN                             07002
         sigclose (catstid.stfile := 0);                                07003
   %search catalogs for document%                                       07004
      IF NOT (catstid ← gtcatent ($docnumber, 0, 0, FALSE)) THEN
                                                                        07005
         BEGIN %search <TEJOURNAL>%                                     07006
         *tempfn* ← "<TEJOURNAL>J", *docnumber*, ".NLS", EOL;           07007
         IF NOT (tempjfn ← sgtjfn (flags, $tempfn, 0)) THEN             07008
            RETURN (NOT thorough);                                      07009
         reljfn (tempjfn);                                              07010
         IF NOT vrprvacc (0, $tempfn, 0, $initsr) THEN                  07011
            err ($"Private document; access denied to you");            07012
         *filename* ← *tempfn*;                                         07013
         RETURN (TRUE);                                                 07014
         END;                                                           07015
   %verify access to document%                                          07016
      IF NOT vrprvacc (0, 0, catstid, $initsr) THEN                     07017
         err ($"Private document; access denied to you");               07018
      acchecked ← TRUE;                                                 07019
   %fetch document's filename%                                          07020
```

```
      IF NOT getcfn (catstid, &filename, 0, 0) THEN                    07021
         err ($"Hardcopy document; not available on-line");            07022
      IF *filename* /filename.L/ # EOL THEN                            07023
         *filename* ← *filename*, EOL;                                 07024
   %close catalog file%                                                07025
      close (catstid.stfile := 0);                                     07026
   %terminate%                                                         07027
      dismes (2, $"Catalog File");                                     07028
      RETURN (TRUE);                                                   07029
      END.                                                             07030
                                                                       07031
(cnvfndocnum) %convert filename to document number%
PROCEDURE (filename, docnumber);                                       06887
   %----------%                                                        06888
   LOCAL TEXT POINTE tp1, tp2;                                        06889
   REF filename, docnumber;                                            06890
   %----------%                                                        06891
   IF NOT FIND SF(*filename*) ('< /'>/ / ) ↑tp1 4$D ↑tp2 ('. ("NLS"
   (EOL/ENDCHR) / ENDCHR / EOL) / EOL / ENDCHR)                        06892
      THEN RETURN (FALSE);                                             06893
   *docnumber* ← tp1 tp2;                                              06894
   RETURN (TRUE);                                                      06895
   END.                                                                06896
                                                                       06897
%.....FTP routines.....%                                               05132
   REF ftperm, ftpoh;                                                  05133
   %FTP service routines%                                              05134
      (ftpbgn) PROCEDURE;                                              05135
         %initialize for FTP activity%                                 05136
         %----------%                                                  05137
         ftpem ← $emsignal;                                            05138
         ON SIGNAL ELSE ftpend ();                                     05139
         &ftperm ← getstring (100, $dspblk);                           05140
         &ftpoh ← getstring (35, $dspblk);                             05141
         ftpcrt ();                                                    05142
         ftpex ($"BGN", 0);                                            05143
         ftpoha ← ftpoh.L ← 0;                                         05144
         ftpsem ($emreset);                                            05145
         RETURN (TRUE);                                                05146
         END.                                                          05147
                                                                       05148
      (ftpend) PROCEDURE;                                              05149
         %end of FTP activity%                                         05150
         %----------%                                                  05151
         ftpem ← $emrtnfalse;                                          05152
         IF ftphnd THEN                                                05153
            BEGIN                                                      05154
            ftpex ($"END", 0);                                         05155
            ftpkil ();                                                 05156
            END;                                                       05157
         IF &ftperm THEN freestring ((&ftperm := 0), $dspblk);         05158
         IF &ftpoh THEN freestring ((&ftpoh := 0), $dspblk);           05159
         RETURN (TRUE);                                                05160
         END.                                                          05161
                                                                       05162
      (ftpopn) PROCEDURE (host);                                       05163
```

```
   %open distant file system%                                          05164
   %----------%                                                        05165
   LOCAL outcome;                                                      05166
   REF host;                                                           05167
   %----------%                                                        05168
   IF (outcome ← ftpex ($"OPEN", 1, &nost)) THEN                       05169
      BEGIN                                                            05170
      ftpoha ← 1; %temp till FTPFRK fixed%                             05171
      *ftpoh* ← *nost*; %temp till FTPFRK fixed%                       05172
      END;                                                             05173
   RETURN (outcome);                                                   05174
   END.                                                                05175
                                                                       05176
(ftpcls) PROCEDURE;                                                    05177
   %close distant file system%                                         05178
   %----------%                                                        05179
   LOCAL outcome;                                                      05180
   %----------%                                                        05181
   IF (outcome ← ftpex ($"CLOS", 0)) THEN ftpoha ← ftpoh.L ← 0;        05182
   RETURN (outcome);                                                   05183
   END.                                                                05184
                                                                       05185
(ftplog) PROCEDURE (user, pass, acct);                                 05186
   %login at distant system%                                           05187
   %----------%                                                        05188
   RETURN (ftpex ($"LOG", 3, user, pass, acct));                       05189
   END.                                                                05190
                                                                       05191
(ftpdml) PROCEDURE (locfile, user);                                    05192
   %send mail to distant user%                                         05193
   %----------%                                                        05194
   RETURN (ftpex ($"DML", 2, user, locfile));                          05195
   END.                                                                05196
                                                                       05197
(ftplml) PROCEDURE (locfile, user);                                    05198
   %send mail to local user%                                           05199
   %----------%                                                        05200
   RETURN (ftpex ($"LML", 2, user, locfile));                          05201
   END.                                                                05202
                                                                       05203
(ftpqml) PROCEDURE (locfile, user, host, stagingdir);                  05204
   %queue mail for user%                                               05205
   %----------%                                                        05206
   RETURN (ftpex ($"QML", 4, locfile, stagingdir, host, user));        05207
   END.                                                                05208
                                                                       05209
(ftpfml) PROCEDURE (locfile, author, authorhost, title, header,
msg);                                                                  05210
   %format mail%                                                       05211
   %----------%                                                        05212
   RETURN (ftpex ($"FML", 6, locfile, author, authorhost, title,
   header, msg));                                                      05213
   END.                                                                05214
                                                                       05215
```

```
(ftpsdd) PROCEDURE (directory);                                    05216
   %set distant default directory%                                 05217
   %----------%                                                    05218
   RETURN (ftpex ($"SDD", 1, directory));                          05219
   END.                                                            05220
                                                                   05221
(ftpelim) PROCEDURE (locfile);                                     05222
   %eliminate local file%                                          05223
   %----------%                                                    05224
   RETURN (ftpex ($"ELIM", 1, locfile));                           05225
   END.                                                            05226
                                                                   05227
(ftpvhst) PROCEDURE (host);                                        05228
   %verify host%                                                   05229
   %----------%                                                    05230
   RETURN (ftpex ($"VHST", 1, host));                              05231
   END.                                                            05232
                                                                   05233
(ftpsto) PROCEDURE (locfile, disfile, xfermode);                   05234
   %send file to distant system%                                   05235
   %----------%                                                    05236
   RETURN (ftpex ($"STOR", 3, disfile, locfile, xfermode));        05237
   END.                                                            05238
                                                                   05239
(ftprtr) PROCEDURE (locfile, disfile, xfermode);                   05240
   %retrieve file from distant system%                             05241
   %----------%                                                    05242
   RETURN (ftpex ($"RETR", 3, disfile, locfile, xfermode));        05243
   END.                                                            05244
                                                                   05245
(ftpapp) PROCEDURE (locfile, disfile, xfermode);                   05246
   %append to file at distant system%                              05247
   %----------%                                                    05248
   RETURN (ftpex ($"APP", 3, disfile, locfile, xfermode));         05249
   END.                                                            05250
                                                                   05251
(ftpdir) PROCEDURE (locfile, remdsg);                              05252
   %retrieve distant directory listing%                            05253
   %----------%                                                    05254
   RETURN (ftpex ($"DIR", 2, remdsg, locfile));                    05255
   END.                                                            05256
                                                                   05257
(ftpdel) PROCEDURE (disfile);                                      05258
   %delete distant file%                                           05259
   %----------%                                                    05260
   RETURN (ftpex ($"DEL", 1, disfile));                            05261
   END.                                                            05262
                                                                   05263
(ftpren) PROCEDURE (curfile, newfile);                             05264
   %rename distant file%                                           05265
   %----------%                                                    05266
   RETURN (ftpex ($"REN", 2, curfile, newfile));                   05267
   END.                                                            05268
                                                                   05269
(ftpsem) PROCEDURE (mode);                                         05270
   %set FTP-primitive error mode%                                  05271
```

```
        %----------%                                                    05272
        ON SIGNAL ELSE                                                  05273
           IF ftpem = $emrtnfalse THEN RETURN (FALSE);                  05274
        CASE mode OF                                                    05275
           = $emrtnfalse, = $emsignal: ftpem ← mode;                    05276
           = $emreset: ftpem ← $emsignal;                               05277
           ENDCASE err ();                                              05278
        RETURN (TRUE);                                                  05279
        END.                                                            05280
                                                                        05281
     (ftprem) PROCEDURE;                                                05282
        %read FTP-primitive error mode%                                 05283
        %----------%                                                    05284
        RETURN (ftpem);                                                 05285
        END.                                                            05286
                                                                        05287
     (ftproh) PROCEDURE (host);                                         05288
        %read open host name and address%                               05289
        %----------%                                                    05290
        REF host;                                                       05291
        IF &host THEN *host* ← *ftpoh*;                                 05292
        RETURN (ftpoha);                                                05293
        END.                                                            05294
                                                                        05295
%FTPFRK driver%                                                         05296
     (ftpex) PROCEDURE (op, count, parm1, parm2, parm3, parm4, parm5,
     parm6);                                                            05297
        %execute FTPFRK operation%                                      05298
        %----------%                                                    05299
        ON SIGNAL ELSE                                                  05300
           IF ftpem = $emrtnfalse THEN RETURN (FALSE);                  05301
        CASE count OF                                                   05302
           IN [0,2]: ftpcf (op, count, parm1, parm2);                   05303
           IN [3,4]:                                                    05304
              BEGIN                                                     05305
              ftpcf ($"ARG", 2, parm1, parm2);                          05306
              ftpcf (op, count-2, parm3, parm4);                        05307
              END;                                                      05308
           IN [5,6]:                                                    05309
              BEGIN                                                     05310
              ftpcf ($"ARG", 2, parm1, parm2);                          05311
              ftpcf ($"ARG", 2, parm3, parm4);                          05312
              ftpcf (op, count-4, parm5, parm6);                        05313
              END;                                                      05314
           ENDCASE ftperr ($"Invalid FTPFRK argument count.");          05315
        RETURN (TRUE);                                                  05316
        END.                                                            05317
                                                                        05318
%fork manipulaton subroutines%                                          05319
     (ftpcrt) PROCEDURE;                                                05320
        %create FTP inferior fork%                                      05321
        %----------%                                                    05322
        LOCAL jfn;                                                      05323
        LOCAL STRING errstr [100];                                      05324
        %----------%                                                    05325
        IF ftphnd THEN ftperr ($"FTP already initialized.");            05326
```

```
    ON SIGNAL ELSE ftpkil ();                                          05327
    IF NOT (jfn ← sgtjfn (getgtjflg (read, 0, oldvrsn),
    $"<NET>FTPFRK.SAV", $errstr)) THEN ftperr ($errstr);               05328
    r1 ← 2511;                                                         05329
    IF NOT SKIP ! JSYS cfork THEN ftperr ($"Can't create inferior
    fork for FTP.");                                                   05330
    ftphnd ← r1;                                                       05331
    ! HRLS 1;                                                         05332
    ! HRR 1,jfn;                                                      05333
    ! JSYS get;                                                       05334
    RETURN;                                                            05335
    END.                                                               05336
                                                                       05337
(ftpkil) PROCEDURE;                                                    05338
    %kill FTP inferior fork%                                           05339
    %----------%                                                       05340
    IF ftphnd THEN                                                     05341
        BEGIN                                                          05342
        r1 ← ftphnd;                                                   05343
        ! JSYS kfork;                                                 05344
        ftphnd ← 0;                                                    05345
        END;                                                           05346
    RETURN;                                                            05347
    END.                                                               05348
                                                                       05349
(ftpcf) PROCEDURE (op, count, parm1, parm2);                           05350
    %invoke FTP inferior fork%                                         05351
    %----------%                                                       05352
    REF op, parm1, parm2;                                              05353
    LOCAL STRING errstr [100];                                         05354
    LOCAL frkacs [16];                                                 05355
    %----------%                                                       05356
    IF NOT ftphnd THEN ftperr ($"FTP not yet initialized.");           05357
    %op code%                                                          05358
        IF op.L > 4                                                    05359
            THEN ftperr ($"FTP op code exceeds four characters.")
                                                                       05360
        ELSE                                                           05361
            BEGIN                                                      05362
            frkacs ← 0;                                                05363
            ftppts (&op, $frkacs);                                     05364
            END;                                                       05365
    %argument count%                                                   05366
        IF count NOT IN [0,2]                                          05367
            THEN ftperr ($"Invalid number of arguments to FTPFRK.")
                                                                       05368
        ELSE frkacs [1] ← count;                                       05369
    %parameter 1%                                                      05370
        IF count > 0 THEN                                              05371
            BEGIN                                                      05372
            IF parm1.L > 34                                            05373
                THEN ftperr ($"FTP parameter exceeds 34
                characters.");                                         05374
            ftppts (&parm1, $frkacs [2]);                              05375
            END;                                                       05376
    %parameter 2%                                                      05377
```

```
      IF count > 1 THEN                                                    05378
         BEGIN                                                             05379
         IF parm2.L > 34                                                   05380
            THEN ftperr ($"FTP parameter exceeds 34
            characters.");                                                 05381
         ftppts (&parm2, $frkacs [9]);                                     05382
         END;                                                              05383
      %start fork%                                                         05384
         r1 ← ftphnd;                                                      05385
         r2 ← $frkacs;                                                     05386
         ! JSYS sfacs;                                                     05387
         r2 ← 0;                                                           05388
         ! JSYS sfrkv;                                                     05389
      %wait for completion%                                                05390
         ! JSYS wfork;                                                     05391
      %check outcome%                                                      05392
         ! JSYS rfsts;                                                     05393
         ! HLRZS 1;                                                        05394
         IF r1 # 2 THEN ftperr ($"FTP inferior fork terminated
         abnormally.");                                                    05395
         r1 ← ftphnd;                                                      05396
         r2 ← $frkacs;                                                     05397
         ! JSYS rfacs;                                                     05398
         IF NOT frkacs [0] THEN RETURN;                                    05399
         ftpgts ($errstr, $frkacs [1]);                                    05400
         ftperr ($errstr);                                                 05401
      END.                                                                 05402
                                                                           05403
%miscellaneous subroutines%                                                05404
   (ftperr) PROCEDURE (msg);                                               05405
      %dispatch FTP error%                                                 05406
      %----------%                                                         05407
      *ftperm* ← *[msg]*;                                                  05408
      SIGNAL ($ftpsig, &ftperm);                                           05409
      END.                                                                 05410
                                                                           05411
   (ftpgts) PROCEDURE (string ,addr);                                      05412
      %convert ASCIZ to NLS string%                                        05413
      %----------%                                                         05414
      REF string;                                                          05415
      LOCAL char, bp;                                                      05416
      bp ← chbmty -1 +addr;                                                05417
      string.L ← 0;                                                        05418
      UNTIL (char ← ↑bp) = 0 DO *string* ← *string*, char;                 05419
      RETURN                                                               05420
      END.                                                                 05421
                                                                           05422
   (ftppts) PROCEDURE (string, addr);                                      05423
      %convert NLS to ASCIZ string%                                        05424
      %----------%                                                         05425
      REF string;                                                          05426
      IF NOT string.L THEN RETURN;                                         05427
      r1 ← chbmty -1 + addr;                                               05428
      r2 ← chbmty + &string;                                               05429
      r3 ← - string.L;                                                     05430
      ! JSYS sout;                                                         05431
```

```
        ! IDPB 3,1;                                                    05432
        RETURN                                                         05433
        END.                                                           05434
                                                                       05435
%Routines dealing with Journal files, directories, etc%                06728
   (jclosfl)PROC;                                                      06729
      %Close any journal files which may be open%                      06730
      IF jworkstid.stfile THEN close(jworkstid.stfile := 0);           06731
      IF jcatstid.stfile THEN close(jcatstid.stfile := 0);             06732
      IF jrnlstid.stfile THEN closeu(jrnlstid.stfile := 0);            06733
      IF diststid.stfile THEN close(diststid.stfile := 0);             06734
      IF numstid.stfile THEN close(numstid.stfile := 0);               06735
      %Set subsystem name back to regular nls%                         06736
         r1 ← nlssbn;                                                  06737
         !JSYS setnm;                                                  06738
      RETURN END.                                                      06739
                                                                       06740
   (jflname)PROC(astr);                                                06741
      %This procedure accepts the name part of a Journal file name as a
      parameter, and constructs a full name from it.                   06742
      The extension is always .NLS, and the user under whom it is
      stored is Journal if it is running as the real system, and duvall
      otherwise (i.e. if jdebug is true)%                              06743
      %The file name is built up in the string 'jnamstr', whose address
      is returned%                                                     06744
      LOCAL STRING tempsr[50];                                         06745
      REF astr;                                                        06746
      *tempsr* ← *astr*;                                               06747
      astruc($tempsr);                                                 06748
      IF *tempsr* = "IDENTFILE" THEN                                   06749
         BEGIN                                                         06750
         IF jdebug THEN *jnamstr* ← "<IDENTFILE>EXP-IDENTS.MASTER"     06751
         ELSE *jnamstr* ← "<IDENTFILE>IDENTS.MASTER";                  06752
         END                                                           06753
      ELSE                                                             06754
         BEGIN                                                         06755
         IF jdebug THEN *jnamstr* ← "<DUVALL>"                         06756
         ELSE *jnamstr* ← "<JOURNAL>";                                 06757
         *jnamstr* ← *jnamstr*, *astr*, ".NLS";                        06758
         END;                                                          06759
      RETURN($jnamstr);                                                06760
      END.                                                             06761

   (conjdir)PROC(cflag);                                               06762
      %IF cflag true, connect to Journal Directory, saving info for
      curren one.   Flag false means connect back%                     06763
      LOCAL byt;                                                       06764
      IF cflag THEN                                                    06765
         BEGIN                                                         06766
         r3 ← byt ← chbptr(0)+$jpassw;                                 06767
         !JSYS gtpsw; %read the password for connected directory%       06768
         jpassw.L ← slngth(byt, r3);                                   06769
         !JSYS gjinf;                                                  06770
         jdirn ← r2; %conncted directory number%                       06771
         r1 ← 1;
         r2 ← chbptr(0) + (IF jdebug THEN $"DUVALL" ELSE $"JOURNAL");
```

```
                                                                    06772
      !JSYS stdir;                                                  06773
      GOTO derr;                                                    06774
      GOTO derr;                                                    06775
      r1.LH ← 0;                                                    06776
      r2 ← chbptr(0) + (IF jdebug THEN $"WSD" ELSE $jnlpsw);        06777
      IF NOT SKIP !JSYS cndir THEN                                  06778
         (derr): err($"Directory Connect Failed");                  06779
      END                                                           06780
   ELSE                                                             06781
      BEGIN                                                         06782
       IF (r1 ← jdirn) = 0 THEN RETURN;                             06783
      r2 ← chbptr(0) + $jpassw;                                     06784
      IF NOT SKIP !JSYS cndir THEN                                  06785
         BEGIN                                                      06786
         !JSYS gjinf;                                               06787
         r2 ← 0;                                                    06788
         IF NOT SKIP !JSYS cndir THEN                               06789
            err($"Connect return failed--left in Journal");         06790
         err($"Connect return failed--returned to Login Directory");
                                                                    06791
         END;                                                       06792
      END;                                                          06793
   RETURN                                                           06794
   END.
                                                                    06795
(conidir)PROC(cflag);                                               06796
   %IF cflag true, connect to identfile Directory, saving info for
   current one.   Flag false means connect back%                    06797
   LOCAL byt;                                                       06798
   IF cflag THEN %connect to <identfile>%                           06799
      BEGIN                                                         06800
      r3 ← byt ← chbmty+$ipassw;                                    06801
      !gtpsw; %read the password for connected directory%           06802
      ipassw.L ← slngth(byt, r3);                                   06803
      !gjinf;                                                       06804
      idirn ← r2; %connected directory number%                      06805
      r1 ← 1;                                                       06806
      r2 ← chbmty + $"IDENTFILE";                                   06807
      !stdir;                                                       06808
      GOTO derr2;                                                   06809
      GOTO derr2;                                                   06810
      r1.LH ← 0;                                                    06811
      r2 ← chbmty + $jnlpsw; %password for <identfile>%             06812
      IF NOT SKIP !cndir THEN                                       06813
         (derr2): err($"Directory Connect Failed");                 06814
      END                                                           06815
   ELSE                                                             06816
      BEGIN                                                         06817
       IF (r1 ← idirn) = 0 THEN RETURN;                             06818
      r2 ← chbmty + $ipassw;                                        06819
      IF NOT SKIP !cndir THEN                                       06820
         BEGIN                                                      06821
         !gjinf;                                                    06822
         r2 ← 0;                                                    06823
         IF NOT SKIP !cndir THEN                                    06824
```

```
            err($"Connect return failed--left in directory
            Identfile");                                          06825
         dismes(2, $"Connect return failed--returned to Login
         Directory");                                             06826
         END;                                                     06827
      END;                                                        06828
   RETURN                                                         06829
   END.

                                                                  06830
(condir)PROC(destname, destpassw, retname, retpassw);             06831
   %Connect to destination directory, and return name and password
   of current directory in indicated strings.                     06832
      Return true if ok, false if not%                            06833
   LOCAL dirno, byt;                                              06834
   LOCAL STRING cdirnam[20], cpassw[20], odirnam[20], opassw[20];
                                                                  06835
   REF destname, destpassw, retname, retpassw;                    06836
   %First get name and password of current directory%             06837
      r3 ← byt ← chbptr(0) + $cpassw;                             06838
      !JSYS gtpsw;                                                06839
      cpassw.L ← singth(byt, r3);                                 06840
   %now get directory name%                                       06841
      !JSYS gjinf;                                                06842
      dirno ← r2;                                                 06843
      byt ← r1 ← chbptr(0) + $cdirnam;                            06844
      r2 ← dirno;                                                 06845
      IF NOT SKIP !JSYS dirst THEN                                06846
         (conderr):                                               06847
         RETURN(FALSE);                                           06848
      cdirnam.L ← singth(byt, r1);                                06849
   %Now connect to new directory%                                 06850
      *odirnam* ← *destname*, 0;                                  06851
      *opassw* ← *destpassw*, 0;                                  06852
      r1 ← 1;                                                     06853
      r2 ← chbptr(0) + $odirnam;                                  06854
      !JSYS stdir;                                                06855
         GOTO conderr;                                            06856
         GOTO conderr;                                            06857
      r1.LH ← 0;                                                  06858
      r2 ← chbptr(0)  * $opassw;                                  06859
      IF NOT SKIP !JSYS cndir THEN GOTO conderr;                  06860
      IF &retname THEN *retname* ← *cdirnam*;                     06861
      IF &retpassw THEN *retpassw* ← *cpassw*;                    06862
   RETURN(1);                                                     06863
   END.                                                           06864
FINISH of loexec                                                  01368
```

(MLK)JNLDEL (MLK)JNLDEL (MLK)JNLDEL (MLK)JNLDEL (MLK)JNLDEL (ML
(MLK)JNLDEL (MLK)JNLDEL (MLK)JNLDEL

```
< NLS, JNLDEL.NLS;352, >, 3-OCT-74 14:38 HGL ;;;;
FILE jnldel % L10 <rel-NLS>jnldel %% (L10,) (rel-nls,jnldel.rel,) %    02
                                                                       03
%Declarations %                                                     01709
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4;                            04
   REF tda, ojsqsw;                                                 01414
   REF rawchr;                                                      04080
%Declarations%                                                      04081
   DECLARE jmaxsmt =  5000;                                         04082
   DECLARE sigdsf = 40215; %discfull signal%                        04166
   DECLARE STRING xlit[2000];                                       03385
% Journal maintenance utility execution %                           03390
   (joutil)PROCEDURE(libf,idstr);                                   03391
      % bits of libf as follows%                                    03392
         % 1B: run recover files %                                  03393
         % 2B: run on-line distribution %                           03394
         % 4B: run slinker %
         % 10B: dettacn, go to sleep, on hour run on-line dist. and
         slinker %                                                  03395
                                                                    03396
         % 20B: Auto Hard Copy%                                     03397
         % 40B: update initial files at on-line delivery%           03398
         % 100B: ask operator for input files to process%           03399
         % 200B: deliver only to idents found in idstr%             03400
         % 400B: ask operator for document numbers to deliver%
         % liblod TRUE: don't set mlav from jdriver file (it was set by
         hand) %                                                    03401
                                                                    03402
      LOCAL curtime, libf1;                                         04160
      % Initialize %                                               03405
         %  Set up operators initials to be XXX %                   03406
            *initsr* ← "XXX";                                       03407
            curtime ← 'X - 100B;                                    03408
            cinit ← 0;                                              03409
            cinit.cint1 ← curtime;                                  03410
            cinit.cint2 ← curtime;                                  03411
            cinit.cint3 ← curtime;
         % set master access print flag; determines whether hard copy
         file should be printed %                                   04161
                                                                    04150
            setmastac();
      % Run recover files,  Check validity of Journal system files
      including content specified in JDRIVER file's RECOVERLIST branch%
                                                                    04109
                                                                    03403
         IF (libf .A 1B) THEN xrecovf();                            04110
      % If nothing else is to be done, then RETURN. %              03404
         IF (libf ← libf .X 1) = 0 THEN RETURN;
      % Turn off recovf bit so recover files will not be done again%
                                                                    04113
                                                                    03427
         libf ← libf .A -2;                                         03412
      IF autostrt THEN                                              03413
         BEGIN                                                      03414
         %Set controlling tty up as primary output again%           03415
            r1 ← 485;                                               03416
            !JSYS gpjfn;                                            03417
            r2.RH ← -1;                                             03418
            r1 ← 485;                                               03419
            !JSYS spjfn;                                            03420
            autostrt←FALSE;                                         03421
         END
```

```
                                                                    03422
ELSE IF (libf .A 10B) THEN                                          04111
   BEGIN                                                            04112
   % Detach and continue processing. %                              03423
   crlf();                                                          03424
   typeas($"Detaching");                                            03425
   !JSYS dtach;                                                     03426
   END;
LOOP % Loop around, processing on instructions from operator the
first time, from the JDRIVER file on successive rounds when
                                                                    03428
awakened. %                                                         03429
   BEGIN
   % Set time table based on values in JDRIVER file; set load
   average cut off if it has not been set already. %                04083
      jsettitab();                                                  03430
   % Distribute on-line documents if requested. %                   04084
      IF (libf .A 2B) THEN                                          03431
         BEGIN                                                      04088
         % Set up JCAT entry. %                                     04085
         % Build distribution file entry. %                        04086
         %Distribute on-line documents%                            04090
            oldist(libf,idstr);                                     04087
         END;                                                       04089
   % Do hard copy distribution printing.  If bit 10B is on, it
   will be run detached with a primary output file set up. %        04091
      IF libf .A 20B THEN  autoncd(libf .A 10B);                    03432
   % Verify and update important files listed in the SLINKERLIST
   branch of JDRIVER.  If things are not OK, lock the Journal
   system. %                                                        04096
      IF libf .A 4B THEN  slinker();                                03433
   % 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. %                                              04097
      IF NOT (libf .A 10B) THEN RETURN;                             03434
      libf←libf .V 16B; %default is on-line deliv and slinker%      03435
                                                                    03436
      r1 ← getsbn($"SNKSLP");                                       03437
      !JSYS setnm; %set up name for sleeping%
      %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%                                                          04098
         timcheck();                                                03438
      r2 ← -1;                                                      03439
      r4 ← 0;                                                       03440
      !JSYS odcnv;                                                  03441
      IF r4 = -1 THEN                                               03442
         r1 ← 1800000 %system does not have time%                   04100
      ELSE                                                          03443
         BEGIN                                                      03444
         curtime ← (r4.RH)/60;                                      03445
         % 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. %                04101
         IF (r1 ← getojtime(curtime + 10 : libfl)) # 0 THEN         03446
            r1 ← r1 - curtime                                       04099
```

```
            ELSE r1 ← 24 * 60 - curtime + getojtime(0 : libf1);  03448
            r1 ← r1 * 60000; %Convert to ms%                     03449
            END;                                                 03450
         !JSYS dismss;                                           03451
         IF libf1 THEN libf ← libf1;                             03452
         END;                                                    03453
      RETURN;                                                    03454
      END.                                                       03455
                                                                 04114
% Verification of Journal system files. %                        03117
   (xrecovf) PROC;
      % "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
                                                                 03116
      OK%                                                        03119
      LOCAL filenc, stid, faterr, recstid;                       03120
      LOCAL TEXT POINTER z1, z2;                                 03121
      faterr ← 0;                                                03122
      ON SIGNAL                                                  03123
         =-5: %We Got here on a badfile, and who knows how%      03124
            IF fileno = bfileno THEN                             03125
               BEGIN                                             03126
               close(fileno := 0);                               03127
               typeas($"File List Bad File");                    03128
               r1 ← 4B5; !JSYS haltf;                            03129
               END                                               03130
            ELSE                                                 03131
               BEGIN                                             03132
               lockjo(1);                                        03133
               typeas($"Bad File--");                            03134
               typeas($lit);                                     03135
               r1 ← 4B5; !JSYS haltf;                            03136
               END;                                              03137
         ELSE                                                    03138
            BEGIN                                                03139
            crlf();                                              03140
            typeas(sysmsg);                                      03141
            crlf();                                              03142
            typeas($"Fatal Error");                              03143
            r1 ← 4B5;                                            03144
            !JSYS haltf;                                         03145
            END;                                                 03146
      lockjo(0); %Lock Journal%                                  03147
      %Detatch and Set up logging TTY 0 as primary output file%  03148
         IF autostrt THEN                                        03149
            BEGIN                                                03150
            crlf();                                              03151
            typeas($"Detaching Slinker");                        03152
            !JSYS dtach;                                         03153
            r1 ← 4B5;
```

```
                                                                    03154
        !JSYS gpjfn;                                                03155
        r2.RH ← 4B5; %tty 0%                                        03156
        r1 ← 4B5;                                                   03157
        !JSYS spjfn;                                                03158
        END;                                                        03159
    r1 ← getsbn($"RECOVF");                                         03160
    !JSYS setnm;                                                    03161
    conjdir(TRUE); %Connect to Journal Directory%                   03162
    enablaccess(0, jrnlaccess);                                     03163
    IF enablW() = -1 THEN                                           03164
       BEGIN                                                        03165
       crlf();
       typeas($"You need WHEEL capability to run recover files");
                                                                    03166
                                                                    03167
       r1 ← 4B5;                                                    03168
       !JSYS haltf;                                                 03169
       END;                                                         03170
    stid ← 0;                                                       03171
    stid.stpsid ← origin;
    IF (stid.stfile ← fileno ← rawopen(jflname($"jdriver"), FALSE)) =
                                                                    03172
    0 THEN                                                          03173
       BEGIN                                                        03174
       typeas($"Driver File open fail");                            03175
       !haltf(4B5);                                                 03176
       END;
    IF (recstid ← namelook(stid, $"recoverlist")) = endfil THEN
                                                                    03177
    snkerr ($"No recoverlist in driver file");                      03178
    stid ← getsub(recstid);                                         03179
    LOOP                                                            03180
       BEGIN                                                        03181
       IF stid = recstid THEN EXIT;                                 03182
       CCPOS SF(stid);                                              03183
       FIND SNP ↑z1 (/NP/ < CH > / SE(stid)) ↑z2;                   03184
       *lit* ← z1 z2;                                               03185
       IF NOT checkfile(jflname($lit), $z2) THEN BUMP faterr;       03186
       stid ← getsuc(stid);                                         03187
       END;                                                         03188
    close(fileno:=0);                                               03189
    %Unlock Journal and proceed if no fatal errors%                  03190
       IF faterr THEN                                               03191
          BEGIN                                                     03192
          typeas($"Fatal File Errors--Journal Left Locked");        03193
          lockjo(1); %just to be sure%                              03194
          !JSYS haltf;                                              03195
          END;                                                      03196
       unlkjo(-1); %allow Journal usage%                            03197
    RETURN;
    END.                                                            03198
                                                                    02965
(slinker) PROC ;
    %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
    three times.  Unlocks journal system if everytning is OK; locks
    it if error detected in one of the listed files; terminates
    through SNKERR if other error.%                                 02966
```

```
                                                                    02967
%-------------------%                                               02968
LOCAL stidl, stid, verfuf, slistid;                                 02969
LOCAL TEXT POINTER z1, z2;                                          02970
stid ← stidl ← 0;                                                   02971
ON SIGNAL                                                           02972
   =-5: %We Got here on a badfile, and who knows how%               02973
      IF stid.stfile = bfilno THEN                                  02974
         BEGIN                                                      02975
         close(stid.stfile  := 0);                                  02976
         snkerr($"File List Bad File");                             02977
         END                                                        02978
      ELSE                                                          02979
         IF stidl.stfile =  bfilno THEN                             02980
            (lockfiles): BEGIN                                      02981
            lockjo(1);
            *lit* ← "Bad File--Probably ", *jnamstr*, EOL, *lit*;
                                                                    02982
                                                                    02983
            snkerr($lit);                                           02984
            END;                                                    02985
   =-6: %I/O data error%                                            02986
      BEGIN                                                         02987
      *lit* ← *[sysmsg]*;                                           02988
      GOTO lockfiles;                                               02989
      END;                                                          02990
   ELSE                                                             02991
      snkerr(sysmsg);                                               02992
r1 ← oljsbn ← getsbn($"SLINKR");                                    02993
!JSYS setnm:                                                        02994
enablaccess(0, jrnlaccess);                                         02995
stid ← 0;                                                           02996
stid.stpsid ← origin;                                               02997
stid.stfile ← open(0, jflname($"jdriver"));                         02999
IF (slistid ← namelook(stid, $"slinkerlist")) = endfil THEN         04092
   snkerr ($"No recoverlist in driver file");                       03000
stid ← getsub(slistid);
verfuf←2; % Verification will take place three times over the
                                                                    03001
list. %                                                             03002
LOOP                                                                03003
   BEGIN                                                            03004
   IF stid = slistid THEN                                           04093
      BEGIN                                                         03005
      IF verfuf THEN                                                03006
         BEGIN                                                      03007
         stid←getsub(stid);                                         03008
         BUMP DOWN verfuf;                                          03009
         END                                                        04094
      ELSE EXIT;                                                    03010
      END;                                                          03011
   IF discfull(400) THEN EXIT;                                      03012
   CCPOS SF(stid);                                                  03013
   FIND $NP ↑z1 ([NP] / SE(stid)) ↑z2;                              03014
   *lit* ← z1 z2;                                                   03015
   stidl ← openlock(0,jflname($lit));                               03016
   crng(TRUE, stidl.stfile);                                        03017
   csdb(TRUE, stidl.stfile);                                        03018
   ckstrc(stidl.stfile);
```

```
                                                                    03019
    IF verfuf THEN                                                  04095
       BEGIN                                                        03020
       updtfl(stidl.stfile, 1, 0);                                  03021
       setaccess(stidl.stfile, jrnlaccess); %reset access%          03022
       END;                                                         03023
    close(stidl.stfile);                                            03024
    stid ← getsuc(stid);                                            03025
    END;                                                            03026
 close(slistid.stfile);                                             03027
 %Now expunge if flag set%                                          03028
    IF flagut(5, $tstfg) THEN                                       03029
       BEGIN                                                        03030
       !JSYS gjinf;                                                 03031
       r1 ← r2; %connected directory number%                         03032
       !JSYS deldf=: %expunge files%                                03033
       END;                                                         03034
 unlkjo(1); %alright to use it now%                                 03035
 RETURN;                                                            03036
 END.                                                               03037

(checkfile) PROC (astr, z2);
    %open the file named in astr, an do anything necessary to assure
    that it is left as an unlocked and legal NLS file%              03038
    %Return false if there was no trace of file, otherwise true%    03039
                                                                    03040
    %-------------------%                                          03041
    LOCAL fileno, retry, dirno, type, stid, jfn;                    03042
    LOCAL TEXT POINTER z1, z3;                                      03043
    LOCAL STRING lookst[50];                                        03044
    REF astr, z2;                                                   03045
    dismes(1, &astr);                                               03046
    retry ← -1;                                                     03047
    (chloop):                                                       03048
    BUMP retry;                                                     03049
    stid ← 0;                                                       03050
    stid.stpsid ← origin;
    IF NOT stid.stfile ← fileno ← rawopen(&astr, TRUE) THEN         03051
    RETURN(0);                                                      03052
    ON SIGNAL                                                       03053
        =-5, =-6: %Bad File or I/O data error%                      03054
          IF (retry = 0) AND [fintadr(fileno)].flpart THEN
             BEGIN %Unlock it, and see if the original is any good%
                                                                    03055
                                                                    03056
             unlkfile(fileno);                                      03057
             dismes(1, $"File Unlocked");                           03058
                %do you want to leave the message up??%             03059
             GOTO chloop;                                           03060
             END                                                    03061
          ELSE                                                      03062
             IF sysgnl = -5 THEN                                    03063
                BEGIN                                               03064
                IF NOT [fintadr(fileno)].flexis THEN
                   IF NOT fileno ← rawopen(&astr, TRUE) THEN
                                                                    03065
                   RETURN(0);                                       03066
                resetf (fileno);                                    03067
                updtfl (fileno, 1, 0);                              03068
                setaccess (fileno, jrnlaccess);
```

```
                                                                    03069
                close (fileno);                                     03070
                dismes (2, $"File Reset");                          03071
                RETURN (0);                                         03072
                END                                                 03073
             ELSE
                BEGIN %try to go back a version on an I/O data error%
                                                                    03074
                                                                    03075
                %Type out a message%
                   dismes(1, $"File Deleted (I/O Data Error)--backed
                   up to previous version");                        03076
                                                                    03077
                %Delete the file%                                   03078
                   IF NOT writeout(fileno, $lit)
                   OR NOT sysclose((jfn ← (fintadr(fileno)).florig :=
                   0) .V %B11) THEN RETURN(0);                      03079
                   delfil(fileno);                                  03080
                   r1 ← jfn;                                        03081
                   IF NOT SKIP !JSYS delf THEN RETURN(0);           03082
                %Now call checkfile again to check the previous
                                                                    03083
                version%                                            03084
                   RETURN(checkfile(&astr, &z2));                   03085
                END;                                                03086
          ELSE;                                                     03087
       cverfil(fileno, 1); %verify file%                            03088
       %do update if there is enough (300 pgs) disc%                03089
          !gdskc(600000777777B);                                    03090
          IF r2 > 300 THEN updtfl(fileno, 1, 0);
       WHILE NOT (FIND z2 $NP ENDCHR) DO %check for content or name%
                                                                    03091
                                                                    03092
          BEGIN                                                     03093
          type ← 0;                                                 03094
          IF FIND z2 "Name: " $NP ↑z1 L $LD ↑z2 ↑z3 THEN            03095
             type ← nametyp                                         03096
          ELSE                                                      03097
             IF FIND z2 "Content: " $NP ↑z1 (';) ↑z2 ↑z3 ←z3 THEN    03098
                type ← contnt                                       03099
             ELSE                                                   03100
                FIND z2  $NP (NP/ENDCHR) ↑z2;                       03101
          IF type THEN                                              03102
             BEGIN                                                  03103
             *lookst* ← z1 z3;                                      03104
             FIND SF(stid) ↑z1;                                     03105
             lookup($z1, $lookst, type);                            03106
             IF z1 = endfil THEN                                    03107
                BEGIN                                               03108
                dismes(1, $"Logical File Error");                   03109
                RETURN(0);                                          03110
                END;                                                03111
             END;                                                   03112
          END;
       setaccess(fileno, jrnlaccess); %Reset access in case it has
                                                                    03113
       changed mystericusly%                                        03114
       close(fileno);                                               03115
       RETURN(TRUE);
       END.                                                         03116
                                                                    03387
% Build distribution file and catalog entries %
```

```
(jinsubs)PROC(flgs, idfilno); %experimental version: takes secondary
distribution requests in tejournal%                                    02313
   %processes submitted items in <TEJOURNAL>%                          02314
   %Bit 100B: go to operator for input files to process%               02315
   LOCAL tstid, dirf, deferd, inftyp, nethcflags:
   local usednum, jcstid, jnlstid, jfn, mesflg, secdflag, recorded;
                                                                       02317
                                                                       02318
   LOCAL TEXT POINTER z1, z2, z3, z4, z5;
   LOCAL STRING tempsr[50], filenm[50], mfname[50], clrkstr[15],
                                                                       02319
   remfname[30];
   LOCAL STRING number[5], modstr[100], dfname[50], dirname[30],
   dayext[10], headstr[80], sourcedir[30] %input%, infstra[36];        02320
                                                                       02322
   jcstid ← jnlstid ← 0;                                               02323
   ON SIGNAL ELSE                                                      02324
      BEGIN                                                            02325
      sigclose(diststid.stfile := 0);                                  02326
      sigclose(jcstid.stfile := 0);                                    02327
      sigclose(jnlstid.stfile := 0);                                   02328
      END;                                                             02559
   CASE jdebug OF                                                      02560
      = TRUE: *sourcedir* ← "<DUVALL>";                                02561
      ENDCASE *sourcedir* ← "<TEJOURNAL>";
   %make up extension "NLn" where n is the last digit of the date%
                                                                       02329
                                                                        02330
      * DATESR* ← NULL;                                               02331
      !JSYS gtad;                                                      02332
      dtfrmt(r1,$datesr);                                              02333
      FIND SF(*datesr*) [1$D] ↑z2 < D ↑z1;                             02334
      *dayext* ← "NL", z1 z2, '; ;                                     02335
   inftyp ← IF jolock() THEN 2 ELSE 3;                                 02336
   jfn ← 0;                                                            02337
   IF inremsiteflag AND NOT discfull(300) THEN                         02338
      %bring across outjournal files from other machine if:%           02339
         %load average is low on both machines%                        02340
         %adequate file space on this machine%                         02341
      BEGIN
      ftprcop (IF lhostn = archost THEN $utilstr ELSE $arcstr,
      $"OUTJOURNAL", $jnlpsw, $dayext, $"<TEJOURNAL>");                02342
                                                                       02343
      END;                                                             02344
   %position distribution stid%
      diststid ← getail(getsub(openlock(0, jflname($"distfile"))));
                                                                       02345
                                                                       02346
   LOOP BEGIN %process input files%                                    02347
      %do tejournal processing if 700 disc pages left%                 02348
         IF discfull(700) THEN EXIT LOOP;                              02349
      %get an NLS file from TEJOURNAL%                                 02350
         IF flgs .A 100B THEN BEGIN %go to user for file name%         02351
            echon();                                                   02352
            crlf();                                                    02353
            typeas($"Input process ");                                 02354
            echoff();                                                  02355
            dirf ← FALSE;                                              02356
            CASE lookc() OF                                            02357
               ='f, ='F, =CA: BEGIN                                    02358
                  input();                                             02359
                  IF lookc() = '< THEN dirf ← TRUE;
```

```
                END;                                                    02360
                                                                        02361
              ='q, ='Q: BEGIN                                           02362
                input();                                                02363
                typeas($"Quit");                                        02364
                %getctc();%                                             02365
                EXIT LOOP;                                              02366
                END;                                                    02367
              ='a, ='A: BEGIN                                           02368
                input();                                                02369
                typeas($"All Files");                                   02370
                %getctc();%                                             02371
                flgs ← flgs .X 100B;                                    02372
                jfn ← 0;                                                02373
                REPEAT LOOP;                                            02374
                END;                                                    02375
              ='<: dirf ← TRUE;                                         02376
              ENDCASE;                                                  02377
            typeas($" File ");                                          02378
            echon();                                                    02379
            *filenm* ← NULL;                                            02380
            IF rdlit($filenm,0) # 1 THEN BEGIN                          02381
              typeas ($"? ");                                           02382
              REPEAT LOOP;                                              02383
              END;                                                      02384
            IF NOT dirf THEN *filenm* ← *sourcedir*, *filenm*;          02385
            echoff();                                                   02386
            IF NOT (jfn←sgtjfn(100101B6,$filenm, $tempsr)) THEN
            REPEAT LOOP;                                                02387
            END                                                         02388
          ELSE BEGIN                                                    02389
            IF NOT jfn THEN CASE (inftyp ← inftyp - 1) OF               02390
              =2: BEGIN %defered%                                       02391
                deferd ← TRUE;                                          02392
                *infstra* ← *sourcedir*, "*.NLP;*";                     02393
                END;                                                    02394
              =1: BEGIN %local numbered%                                02395
                deferd ← FALSE;                                         02396
                *infstra* ←  *sourcedir*, "*.NLN;*";                    02397
                END;                                                    02398
              =0: BEGIN %remote numbered%                               02399
                deferd ← FALSE;                                         02400
                *infstra* ← *sourcedir*, "*.NLR;*";                     02401
                END;                                                    02402
              ENDCASE EXIT LOOP;
            IF NOT (jfn←sgtjfn(100101B6, $infstra, $tempsr)) THEN
            REPEAT LOOP;                                                02403
            END;                                                        02404
        % rename  with "NLD" extension%                                 02405
          jfn ← jrnamf(jfn,$filenm,$"NLD;");                            02406
        %open, etc%                                                     02407
          jworkstid.stfile←open(jfn,$filenm);                           02408
          jworkstid.stpsid←origin;                                      02409
          makeptr(jworkstid, $jwp1);                                    02410
        % set number, mesflg,  %                                       02411
          rfcflg←FALSE;                                                 02412
          IF NOT (FIND jwp1 ([/"(J" / "(j"/ ↑z2 ↑z1 ) ←2z1 4$5 D ↑z3)
```

```
                                                                    02413
THEN BEGIN                                                          02414
   %bad file, rename with "BAD as ext.%                             02415
   (olbadf): close (jworkstid.stfile:=0);                           02416
   reljfn(jrnamf(0,$filenm,$"BAD;"));                               02417
   REPEAT LOOP;                                                     02418
   END;                                                             02419
*number*←z2 z3;                                                     02420
IF flgs .A 100B THEN deferd ← (*number* = "1234");                   02421
IF NOT deferd AND *number* = "1234" THEN GOTO olbadf;               02514
secdflag ← FALSE;                                                   02636
% set message flag %                                               02448
   FIND z1 < CH;                                                    02449
   CASE READC OF                                                    02450
    .  ='H: mesflg←hcopyv;                                          02451
       ='F, ='M: % let size determine type %                        02790
          mesflg ← jtypchk(2000, jworkstid);                        02515
       ='U: secdflag ← 2;                                           02516
       ='S: secdflag ← 1;                                           02453
       ENDCASE %error%;                                             02639
% set recording flag %                                             02758
   FIND z1;                                                         02629
   CASE READC OF                                                    02630
      ='U: recorded ← FALSE;                                        02635
      ENDCASE                                                       02631
         recorded ← TRUE;                                           04379
%make sure there are 4 semicolons before z1%                        04380
   IF (FIND z1 < [';] ↑z4) AND (NOT FIND ";;;") THEN ST z4          02456
   z4 ← ":;;";                                                      02457
%open tjcat if it hasn't been already%                              02517
   IF recorded AND NOT jcstid.stfile THEN jcstid ←                  02521
   openlock(0, jflname($"tjcat"));                                  02519
IF secdflag = 2 THEN                                                02520
   BEGIN                                                            02516
   %jcat entry not here yet, try to get it%                         02523
   %not implemented yet%                                            02525
   END;                                                             02524
IF secdflag                                                         02526
   THEN nethcflag ← getnhcflag(inftyp # 0)
   ELSE
      BEGIN
      FIND jwpl > ([*clerk: *] [L/D]) ↑z4 ←z4 $(L/D) ↑z5;           02422
                                                                    02423
      *clrkstr* ← z4 z5;                                            02424
      %get delivery flag for clerk on this machine%                 02531
         %don't use this for now                                    02425
         ckident($clrkstr, $xlit, idfilno);                         02426
         getinlhost($xlit, 0, $z4, $z5);                            02427
         nethcflag ← CASE lhostn OF                                 02428
            = archost:
               IF (FIND BETWEEN z4 z5 ([*arcstr*])) OR
               NOT (FIND BETWEEN z4 z5 ([*utilstr*]))
               THEN TRUE ELSE FALSE;                                02429
            = utilhost:                                             02430
               IF FIND BETWEEN z4 z5 ([*utilstr*]) THEN
               TRUE ELSE FALSE;                                     02431
            ENDCASE 0;                                              02432
```

```
                %                                                   02532
                nethcflag ← getnhcflag(inftyp # 0);                 02534
             END;                                                   02527
      IF deferd THEN BEGIN                                          02433
         IF *number* # "1234" THEN GOTO olbadf;                     02434
         getcnum($clrkstr, $number, $" JOURNAL ", 2, 0);            02435
         ST z2 z3 ← *number*;                                       02436
         FIND jwp1 > ([".HJOURNAL="] ["1234"]) ↑z3 < 4CH ↑z2;
                                                                    02437
         ST z2 z3 ← *number*;                                       02438
         END;                                                       0243
      IF outremsiteflag THEN                                        02440
         BEGIN                                                      02441
         *remfname* ← "<OUTJOURNAL>R", *number*, ".NLR;";           02442
         IF inftyp # 0 AND NOT updtfl(jworkstid.stfile, 1,
         $remfname ) THEN snkerr ($"Journal Remote Update
         Error");                                                   02443
         END;                                                       02444
      IF NOT secdflag THEN                                          02445
         BEGIN                                                      02529
         setjheader($number, $modstr, idfilno); %fill out journal
         header statement in tejournal or outjournal%               02528
         FIND jwp1 > ([("(J" / "(j"] ↑z1 ↑z3 ) ←2z1 ←4z3  [';]
         ↑z2;                                                       02513
         *headstr* ← z1 z2;                                         02541
         ST jworkstid ← SF(jworkstid) z3, z2 SE(jworkstid);         02455
         END;                                                       02530
   IF secdflag THEN dsecdoc($number, nethcflag, inftyp#0, jcstid,
   idfilno)                                                         02535
   ELSE CASE mesflg OF                                              02536
      = 1: BEGIN %message%                                          02458
         CASE recorded OF                                           02640
            = TRUE:                                                 02641
               BEGIN                                                02642
               IF NOT (jsavaccess ← access ,A
               accmask[jrnlaccess]) THEN                            02460
                  enablaccess(0, jrnlaccess);                       02461
               IF jnlstid.stfile THEN usednum ← usednum + 20 ELSE
               jnlstid ← findactive(20, jcstid: usednum); %number
               should be number of statements in prospective
               journal document%                                    02462
               entrjdoc(jnlstid, $number,$headstr);                 02463
               updjcat($number, usednum, $mfname, jcstid);
               %update jcat file%                                   02464
               *tempsr* ← 'J, *number*;                             02465
               jcatentry(IF jdebug THEN $"DUVALL" ELSE
               $"JOURNAL", $mfname,  %file name%  $tempsr, FALSE,
               jcstid,$headstr); %branch name%                      02466
               %Now insert Catalog Modification Commands as
               necessary%                                           02467
                  IF modstr.L > 0 THEN                              02468
                     BEGIN                                          02469
                     IF (tstid ← namelook(jcstid, $"Catmods")) =
                     endfil THEN                                    02470
                        BEGIN                                       02471
                        IF (tstid ← namelook(jcstid,
```

```
                    $"jcatalog")) = endfil THEN
                    badfil(jcstid.stfile);                          02472
                    tstid ← cis(tstid, $"(catmods) Catalog
                    Modifiction Commands", $sucdir);                02473
                    END;                                            02474
                 cis(tstid, $modstr, $downstr);                     02475
                 END;                                               02476
              END;                                                  02643
           ENDCASE % unrecorded message %                           02644
              BEGIN                                                 02767
              % assuming work file extension = .NLx %               02768
              FIND SF(*filenm*) ↑z1;                                02769
              IF FIND > z1 /(".NL")/ ↑z2 CH ↑z3 THEN                02770
                 *mfname* ← z1 z2, 'U, z3 SE(*filenm*)              02771
              ELSE  *mfname* ← *filenm*;                            02772
              END;                                                  02773
        distdoc($mfname, IF jdebug THEN $"DUVALL" ELSE
        $"JOURNAL", $number, TRUE, $headstr, idfilno, nethcflag,
        recorded); %distribute it%                                  02477
        END;                                                        02478
     ENDCASE BEGIN %non message--mesflg#1%                          02479
        CASE recorded OF                                            02645
           = TRUE:                                                  02649
              BEGIN                                                 02646
              IF NOT (jsavaccess ← access .A
              accmask/jrnlaccess/) THEN                             02480
                 enablaccess(0, jrnlaccess);                        02481
              %Now open a work file with number as name%            02482
                 IF mesflg # hcopyv THEN                            02483
                    BEGIN                                           02484
                    jdocfile(jcstid, $number, $jworkstid,
                    $dirname); %Update to new file in proper
                    directory%                                      02485
                    END;                                            02486
              %Now update control file%                             02487
                 jcatentry($dirname, $number, $"1", mesflg,
                 jcstid, $headstr);                                 02488
                 %Now insert Catalog Modification Commands as
                 necessary%                                         02489
                 IF modstr.L > 0 THEN                               02490
                    BEGIN                                           02491
                    IF (tstid ← namelook(jcstid, $"Catmods"))
                    = endfil THEN                                   02492
                       BEGIN                                        02493
                       IF (tstid ← namelook(jcstid,
                       $"jcatalog")) = endfil THEN
                       badfil(jcstid.stfile);                       02494
                       tstid ← cis(tstid, $"(catmods) Catalog
                       Modifiction Commands", $sucdir);             02495
                       END;                                         02496
                    cis(tstid, $modstr, $downstr);                  02497
                    END;                                            02498
                 *dfname* ← *number*;                               02499
              END;                                                  02647
        ENDCASE % unrecorded file %                                 02648
           BEGIN                                                    02759
```

```
                % NOT IMPLEMENTED - NEED STORAGE LOCATION, agreements
                with archive system %                                   02774
                IF jdebug THEN err($" xnls journal system error");
                                                                        02775
                                                                        02765
                END;
              distdoc($dfname, $dirname, $number, mesflg, $headstr,
              idfilno, nethcflag, recorded); %distribute it%            02500
              END;                                                      02501
          delpc(jworkstid.stfile,$tempsr);                              02502
          close(jworkstid.stfile := 0);                                 02503
          %rename with ext reflecting the date and release jfn%         02504
             reljfn(jrnamf(0,$filenm, IF recorded THEN $dayext ELSE
             $"NLU;"));                                                 02505
                                                                        02506
          END;                                                          02507
       %Close distfile, tjcat, message file%                            02508
          close (diststid.stfile := 0);                                 02509
          close (jcstid.stfile := 0);                                   02510
          closeu(jnlstid.stfile:=0);
       RETURN END.                                                      02512

% Process distribution file-- online distribtion %                      03388
   (oldist)PROC(flgs,idstr);                                              022
      %Does the on-line distribution of JOurnal Documents%                023
      %Bit 40B of flgs to update users' initial files at delivery%       01147
      %Bit 100B: go to operator for input files to process%              01478
      %Bit 200B: only deliver to idents listed in idstr (may contain
      group idents)%                                                    01479
      %Bit 400B: ask operator for document number to distribute%        01519
      LOCAL stid, unrecstid, stidl, tstid, idstid, pcap, adrstid, nost,
      hcdstid, jautstid, jdelstid, jinfostid, xstr, wrkstr;               024
      LOCAL flguif, wierdlock, dnetflag, dolflag, dlinkflag, mailoflag,
      recorded, message, actflag, adstr[40], mhstid;                      025
      LOCAL TEXT POINTER z1, z2, z3, z4, z5, idstptr;                     026
      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];                    027
      REF idstr, xstr, wrkstr;                                            029
      %Set up subsy name%                                                 030
         r1 ← oljsbn ← getsbn($"OLJDEL");                                 031
         !JSYS setnm;                                                     032
      %update users' inital files?%                                     01148
         flguif← flgs .A 40B;                                           01149
      %Enable access to Journal Files%                                    033
         enablaccess(0, jrnlaccess);                                      034
      %Set up signal and initialise parms%                                035
         diststid ← hcdstid ← ctlstid ← &xstr ← &wrkstr ← mailoflag ←
         0;                                                               036
         ON SIGNAL ELSE                                                   037
            BEGIN                                                         038
            sigclose(hcdstid.stfile := 0);                                040
            sigclose(idstid.stfile := 0);                                 041
            close(jrnlstid.stfile := 0);                                  043
            closeu(ctlstid.stfile := 0);                                  044
            IF &xstr # 0 THEN freestring(&xstr:=0,$dspblk);             01728
            IF &wrkstr # 0 THEN freestring(&wrkstr:=0,$dspblk);         02940
            END;                                                          045
```

```
%open id file%                                                          01155
   idstid ← orgstid;                                                    01153
   idstid.stfile ← open(0, jflname($"identfile"));                      01152
%process tejournal%                                                     01724
   jinsubs(flgs, idstid.stfile#);
%Update distribution file%                                               0134
   IF updhcdist() = 0 THEN                                               0135
      BEGIN                                                              0136
      %Distribution file empty%                                          0137
      rl ← nlssbn;                                                       0138
      !JSYS setnm;                                                       0139
      RETURN;                                                            0140
      END;                                                               0141
%If doing restricted distribution, expand the distribution list
idents%                                                                 01476
   IF flgs .A 200B THEN iexpdist(&idstr, idstid.stfile);                01477
   FIND SF(*idstr*) †idstptr;                                           01480
%If doing one document then get number from operator%                   01506
   (oldsdn): IF flgs .A 400B THEN BEGIN                                 01507
      typeas($"Single document number: ");                              01508
      %get number%                                                      01509
      CASE lookc() OF                                                   01511
         IN ['1, '9]: BEGIN                                             01513
            %tnumber($jnumstr);%                                        01520
            *jnumstr* ← 'J, *jnumstr*;                                  01517
            END;                                                        01515
         ='X, ='x: BEGIN                                                01514
            inpcuc(); %pass the "x"%                                    01525
            typeas($" Do all documents now?");                          01524
            IF inpcuc() # CA THEN GOTO oldqui;                          01526
            flgs ← flgs .X 400B;                                        01518
            END;                                                        01516
         ='Q: BEGIN                                                     01527
            typeas($"Quit");                                            01528
            GOTO oldqui;                                                01529
            END;                                                        01530
         ENDCASE GOTO oldsdn;                                           01512
      END;                                                              01510
%Now open distribution file%                                             0142
   hcdstid ← getsub(openlock(0, jflname($"hcdistfile")));               0143
%Initialise the done string (xstr)%                                      0144
   &xstr ← getstring(2000, $dspblk);                                     0145
% Allocate work string. %                                              02941
   &wrkstr ← getstring(2000, $dspblk);                                  02942
*jrnlfnm* ← NULL; %name of currently open message file%                 02015
unrecstid ← 0; % temp file in <tejournal>, location of unrecorded
message %                                                               02812
%Now look for people to send stuff to%                                   0149
   LOOP % over documents in hcdistfile %                                 0150
      BEGIN                                                              0151
      IF hcdstid.stpsid = origin OR discfull(500) THEN                   0152
         % hcdistfile is empty or there are no more documents to
         process %                                                      04186
         EXIT;                                                          04187
      ON SIGNAL                                                         02950
         =sigdsf: EXIT LOOP;                                            02951
```

```
                                                                    02952
      ELSE                                                          02953
         BEGIN
         IF NOT flgs .A 200B THEN hcdstid ← getsuc(hcdstid);
                                                                    02954
                                                                    02955
         IF flguif OR wierdlock THEN                                02956
            BEGIN                                                   02957
            pcap ← enablw();                                        02958
            closeu(ctlstid.stfile := 0);                            02959
            IF pcap # -1 THEN disablw(pcap);                        02960
            END                                                     02961
         ELSE close(ctlstid.stfile := 0);                           02962
         IF mailoflag THEN closemail();                             02963
         REPEAT LOOP;                                               02964
         END;                                                       01493
   IF flgs .A 400B THEN                                             04189
      BEGIN
      % deliver only to documents specified by operator; check
      if this is the one we should be working on %                  04190
      CCPOS SF(hcdstid);                                            01498
      *tempsr* ← NULL;                                              01499
      xtrnam(.tempsr, $swork, -1);                                  01500
      IF *tempsr* # *jnumstr* THEN                                  01501
         BEGIN                                                      04191
         hcdstid ← getsuc(hcdstid);                                 01503
         REPEAT LOOP;                                               01504
         END;                                                       01502
      typeas($" Found");                                            01505
      END;                                                          01494
   adrstid ← getsub(hcdstid); % Get the first addressee branch
   of this document. %                                              0160
   LOOP % over all addressees in this currently prime
   document. %                                                      0161
      BEGIN                                                         0162
      IF flgs .A 200B THEN                                          01481
         BEGIN %deliver only to idents in idstptr%                  04192
         IF NOT FIND idstptr $NP ↑z1 1$LD ↑idstptr THEN EXIT;
                                                                    01485
            % no more idents had been specified by the
            operator%                                               04193
         *fnsr* ← z1 idstptr;                                       01486
          ADRSTID < HCDSTID;                                        01487
         lookup($adrstid, $fnsr, seqname);                          01488
            % Look for things to be deliverd to this user; if
            none, go on to the next. %                              04194
         IF adrstid = endfil THEN REPEAT LOOP;                      01489
         END                                                        01482
      ELSE                                                          01483
         BEGIN                                                      04195
         IF adrstid = hcdstid THEN EXIT;                            0163
            % There are no more addressees for this document;
            go on to the next document. %                           04196
         CCPOS SF(adrstid);                                         0164
         *fnsr* ← NULL;                                             0165
         xtrnam($fnsr, $swork, -1);                                 0166
         END;                                                       01484
      % At this point, FNSR has the ident of the user whose
```

```
mail is beig processed. %                                          04197
IF flgs .A 400B AND getup(adrstid) # hcdstid THEN                  01492
   EXIT LOOP; %only deliver hcdstid branch%                        04198
IF discfull(400) THEN EXIT 2;  %mainly to check load
                                                                   01741
average%                                                            0167
IF fnsr.L # 0                                                       0168
   AND (NOT FIND SF(*xstr*) [SP *fnsr* SP])                         0169
   AND  ckident($fnsr, $xlit, idstid.stfile)
   AND  ((dolflag ← donline(adrstid :actflag)) .V                   0170
   (dnetflag ← dnetdel(adrstid)) # 0) THEN                          0171
      BEGIN  %Try to distribute it%                                01978
      IF dnetflag THEN                                             01980
         BEGIN                                                     01982
         getihost($xlit, $hoststr, 0, 0);                          01983
         getinma($xlit, $dnetma, 0, 0);                            01981
         END;
      *xstr* ← *xstr*, *fnsr*, SP; %mark him as being
                                                                    0172
      tried%                                                        0173
      *tinit* ← *fnsr*; %save off initials%
      %Now see if we can open his control (initial)
                                                                    0174
      file%                                                         0175
         luser($xlit, $tempsr, 0, 0);
         *user* ← *tempsr*; %save off his username %
                                                                   02813
         IF dolflag AND NOT (ctlstid ← opnctfile($fnsr,
         $tempsr : wierdlock%locked by user flag%)) THEN
                                                                   02085
                                                                   04199
            dolflag ← FALSE;                                       02087
         IF dnetflag THEN
            dnetflag ← mailoflag ← openmail($dnetma,
            $hoststr, $"DOCUMENTATION");                           04200
         IF NOT dolflag AND NOT dnetflag THEN                      04201
            GOTO nodo; % no online delivery necessary. %
                                                                   04202
         %By here, ctl file and/or sequential file is
                                                                    0177
         open%                                                      0178
         jautstid ← jdelstid ← jinfostid ← 0;                       0179
      %Now start to give him the documents%
         stid ← adrstid; % so we can get the next user
                                                                    0180
         to be delivered to %                                       0181
         LOOP % over all documents to this user. %                  0182
            BEGIN
            %Get stid of branch in initial file to put
                                                                    0183
            it in%                                                 01989
               IF dolflag THEN                                     04203
                  BEGIN
                  IF FIND SF(stid) ["Author Copy"] THEN
                                                                    0184
                                                                    0185
                     BEGIN                                          0186
                     IF jautstid = 0 THEN
                        IF (jautstid ← namelook(ctlstid,
                        $"author")) = endfil THEN                   0187
                           jautstid ← cis(ctlstid,
                           $"(author) Journal documents
                           authored", $sucdir);                     0188
                     ctlstid ← jautstid;                            0189
```

```
        END                                                    0190
      ELSE                                                     0191
        BEGIN                                                  0192
        dolflag ← dolflag .V 2; %flag for
        stuff to "Journal" branch%                             02162
        CASE actflag OF                                        02548
          = TRUE:                                              02549
            BEGIN                                              02550
            IF jdelstid = 0 THEN                               0193
              IF (jdelstid ←
              namelook(ctlstid,
              $"Action")) = endfil THEN
                                                               0194
              jdelstid ← cis(ctlstid,
              $"(Action) Journal
              documents (most recent
              first)", $sucdir);                               0195
            ctlstid ← jdelstid;                                0196
            END;                                               02551
          ENDCASE                                              02552
            BEGIN                                              02553
            IF jinfostid = 0 THEN                              02554
              IF (jinfostid ←
              namelook(ctlstid,
              $"Info")) = endfil THEN
                                                               02555
              jinfostid ← cis(ctlstid,
              $"(Info) Journal documents
              for information only (most
              recent first)", $sucdir);
                                                               02556
            ctlstid ← jinfostid;                               02557
            END;                                               02558
        END;                                                   0197
      END;                                                     01990
stidl ← getup(stid); % head of branch%                         0198
%Set up link/location pointers%                                02666
  dlinkflag ← FALSE;                                           02667
  recorded ← TRUE;                                             02668
  message ← FALSE;                                             02776
  CCPOS SF(stidl);                                             02669
  IF FIND ["Hard Copy--Location: "] ↑z1
  [';] ↑z2 THEN                                                02670
    GOTO uselnk                                                04204
  ELSE FIND [')] ['(] ↑z1 ←z1 [')] ↑z2;
  %Pointers to link%                                           02671
  dlinkflag ← TRUE;                                            02672
  lnkpspc(1, $z1, $tempsr, $filenm, $stat,
  $spcvw, $adstr);                                             02673
  CASE *stat*[1] OF                                            02674
    = 'J:                                                      02675
      *filenm* ← *filenm*, ",NLS";                             02676
    ENDCASE recorded ← FALSE;                                  02677
  IF FIND BETWEEN z1 z2                                        02678
    ("JRNL"/"DUVALL")%experimental system
    only%) THEN                                                02779
```

```
BEGIN                                                    02679
message ← TRUE;                                          02778
IF recorded THEN                                         02680
   BEGIN                                                 02681
   IF *filenm* # *jrnlfnm* THEN                          02682
      BEGIN                                              02808
      close(jrnlstid.stfile := 0);                       02683
      jrnlstid ← openlock(0, $filenm);                   02684
      *jrnlfnm* ← *filenm*;                              02685
      END;                                               02686
   END                                                   02691
ELSE % unrecorded %                                      02692
   BEGIN                                                 02693
   unrecstid ← orgstid;                                  02791
   unrecstid.stfile ← open(0,
   $filenm);                                             02694
   FIND SF(unrecstid) ↑z5;                               02695
   z4 ← getsub(z5); % location of
   message plex %                                        02792
   GOTO uselnk;                                          02809
   END;                                                  02696
FIND SF(jrnlstid) ↑z3;                                   02687
specreg($stat, nametyp, $z3);                            02688
IF z3 = endfil THEN GOTO uselnk;                         02689
z4 ← getsub(z3); % location of message
plex %                                                   02810
dlinkflag ← FALSE;                                       02690
IF NOT olhed(stid, message, z3, $z1,
$z2, recorded, &wrkstr) THEN                             02698
   GOTO uselnk; %format citation into
   lit (and wrkstr if there is a
   comment). go elsewhere if
   unsuccessful%                                         04205
END                                                      02699
ELSE                                                     02700
   BEGIN %Put out the link%                              02701
   % handle unrecorded files here %                      02750
   (uselnk):                                             02702
   %format citation into lit (and wrkstr
   if there is a comment%                                02703
      olhed(stid, message, z1, $z1, $z2,
      recorded, &wrkstr);                                02704
   END;                                                  02705
%put it out%                                             02706
IF dolflag THEN                                          02707
   BEGIN                                                 04206
   % citation %                                          04220
      tstid ← cis (ctlstid, $lit,
      $downstr);                                         02708
   IF message AND (z3 # endfil) THEN                     02709
      % message %                                       04221
      BEGIN                                              04376
      mhstid ← cis(tstid, $"Message:",
      $downstr);                                         04378
```

```
                ccopgro (mhstid, down, z4,
                getail(z4), 0, $" ");                    02720
                END;                                     04377
             IF wrkstr.L > 0 THEN                        02710
                % comment %                              04222
                cis(tstid, &wrkstr, $downstr);           04207
             IF unrecstid.stfile THEN                    02811
                close(unrecstid.stfile:=0); % temp
                location of unrecorded messages %
                                                         04208
             END;                                        02711
          IF dnetflag THEN                               02712
             BEGIN                                       04209
             IF dlinkflag AND gtollink($z1, $z2)
             THEN                                        02713
                BEGIN                                    04210
                %make it more like network%              02714
                   host ← IF lhostn = archost THEN
                                                         04223
                      $"SRI-ARC" ELSE $"OFFICE-1";
                                                         04224
                   ST z1 z2 ← *[host]*, SP,
                   *filenm*, ";xnls";                    02715
                END;                                     02716
             mailstring($lit, 1);                        02717
             IF wrkstr.L > 0 THEN                        04211
                mailstring(&wrkstr, 2);                  04212
             END;                                        02719
       %Now fix up hcdistfile %                           0227
          IF dolflag AND FIND SE(stid) < ['*] > ["
          do-ola"/"do-ol"] ↑z2 < [SP] ↑z1 THEN           02071
             ST z1 z2 ← " ol-done";                      04213
          IF dnetflag AND FIND SE(stid) < ['*] > ["
          do-net"] ↑z2 < [SP] ↑z1 THEN                   02074
             ST z1 z2 ← " net-done";                     04214
       %Now proceed to next document for this user%
                                                         0229
          LOOP                                           0230
             BEGIN                                       0231
             z1 ← stid;                                  0232
             lookup($z1, $tinit, seqname);               0233
             IF z1 = endfil THEN EXIT 2; %done with
             this user; may have other documents
             for other users to process%                 0234
             stid ← z1;                                  0235
             CCPOS SF(z1);                               0236
             IF donline(stid : actflag)  THEN EXIT;
             ?/Do this one%                               0237
             END; %of search loop%                       0238
       END; %of main dstribution loop for a user%
                                                         0239
    %Now close his control file%                         0240
       IF dolflag THEN                                   02018
          BEGIN                                          04215
          IF (dolflag .A 2) THEN %mark fdb with
          "journal mail" flag%                           02157
```

```
                                BEGIN                                   02164
                                pcap ← enablw();                        02816
                                r1 ← 24B6 +
                                !fintadr(ctlstid.stfile)!.florig;       02158
                                r2 ← 0;                                 02159
                                r2.ojdelf ← r3.ojdelf ← 1;              02160
                                !JSYS chfdb;                            02161
                                *notify* ← *tinit*, ": You have new
                                Journal mail";                          02814
                                xsndtmsg($notify, $user);               02815
                                IF pcap # -1 THEN disablw(pcap);        02817
                                END;                                    02163
                             IF flguif %update initials files%  OR
                             wierdlock THEN                             01160
                                BEGIN                                   01161
                                pcap ← enablw();                         0241
                                closeu(ctlstid.stfile := 0);             0242
                                IF pcap # -1 THEN disablw(pcap);         0243
                                END                                     01162
                             ELSE close(ctlstid.stfile := 0);           01164
                             END;                                       02019
                          IF dnetflag THEN                              02021
                             BEGIN                                      02152
                             closemail();                               02151
                             mailoflag ← 0;                             02205
                             % set up mailer (?) for network
                             distribution. %                            04140
                                pokemailer($"DOCUMENTATION");           02154
                             END;                                       02153
                       END; %of this user%                               0244
                 (nodo): IF NOT flgs .A 200B THEN                        0245
                    % Get the next addressee for the currently prime
                    document and try to process all of his mail. %      04217
                    adrstid ← getsuc(adrstid);                          04216
                 END; %Of this document%                                 0246
              IF NOT flgs .A 600B THEN                                   0247
                 % get the next document to be processed %               04219
                 hcdstid ← getsuc(hcdstid);                             04218
              END; % of loop over documents in hcdistfile. %             0248
           IF flgs .A 400B THEN                                         01521
              GOTO oldsdn; % Documents to be delivered were specified by
              operator; go back to see if there were more specified. %
                                                                        04186
        %Now wrap it up%                                                 0249
                                                                        01523
           (oldqui):                                                     0250
           close(hcdstid.stfile := 0);                                   0251
           close(idstid.stfile := 0);                                    0252
           close(jrnlstid.stfile := 0);                                 02016
           *jrnlfnm* ← NULL;                                             0253
           closeu(ctlstid.stfile := 0);                                 01726
           freestring (&xstr, $dspblk);                                 02943
           freestring (&wrkstr, $dspblk);                                0256
        RETURN
        END.                                                             0257

% Process distribution file-- offline hardcopy distribtion %             04143
```

```
(autohcd)PROC(rundet);                                                  0419
   %Run hard copy automatically.  If rundet is TRUE, set up primary
   output file and run detached.%                                       0420
   LOCAL det, pjfn;                                                     0421
   LOCAL TEXT POINTER z1, z2;                                           0422
   LOCAL STRING tempsr[50];                                             0423
   %Initialise parms and set up SIGNAL%                                 0424
      pjfn ← det ← 0;                                                   0425
      ON SIGNAL ELSE                                                    0426
         BEGIN                                                          0427
         %check for disc full termination%                              02194
            CASE sysgnl OF                                              02192
               = sigdsf: BEGIN                                          02190
                  dismes (1, sysmsg);                                   02195
                  jdquit();                                             02197
                  END;                                                  02196
               ENDCASE;                                                 02193
         %Now type message and restore prinary ouput file to normal%    0428
            IF rundet  AND pjfn THEN                                    0429
               BEGIN                                                    0430
               %Fix up primary output %                                 0431
                  r1 ← 4B5;                                             0432
                  !JSYS gpjfn;                                          0433
                  r2.RH ← r2.LH;                                        0434
                  !JSYS spjfn;                                          0435
               sysclose(pjfn := 0, 0);                                  0436
               END;                                                     0437
            %type a message indicating automatic stop%                  0438
               *tempsr* ← NULL;                                         0439
               !JSYS gtad;                                              0440
               dtfrmt(r1, $tempsr);                                     0441
               *tempsr* ← *tempsr*, " Auto Hard Copy stop";             0442
               IF det  = -1 THEN %Running detatched...type out
               message to logging tty%                                  0443
                  specttyout(0, $tempsr);                               0444
         RETURN;                                                        0445
         END;                                                           0446
   %First type a message indicating automatic startup%                  0447
      *tempsr* ← NULL;                                                  0448
      !JSYS gtad;                                                       0449
      dtfrmt(r1, $tempsr);                                              0450
      *tempsr* ← *tempsr*, " Autc Hard Copy Start";                     0451
      !JSYS gjinf;                                                      0452
      IF (det ← r4) = -1 THEN %Running detatched...type out message
      to logging tty%                                                   0453
         specttyout(0, $tempsr)                                         0454
      ELSE typeas($tempsr);                                             0455
   %operator is "XXX"%                                                  02200
      *cpstr* ← "XXX";                                                  02201
   %Close initial file%                                                 0456
      close(fda).dacsp.stfile:= 0);                                     0457
   %Open file for primary output%                                       0458
      IF rundet THEN                                                    0459
         BEGIN                                                          0460
         *tempsr* ← NULL;                                               0461
```

```
            !JSYS gtad;                                                    0462
            dtfrmt(rl, $tempsr);                                           0463
            IF FIND SF(*tempsr*) /SP/ ↑z1 ←z1 SE(z1) ↑z2  THEN             0464
               ST z1 z2 ← NULL;                                            04144
            *tempsr* ← *tempsr*, "autohcd.out";                            0465
            pjfn ← setupop(0, $tempsr, 4, IF det = -1 THEN 0 ELSE -1);
                                                                           0466
            END;                                                           0467
         %Now set up and run hard copy%                                    0468
            inithcd();                                                     0469
            printall(1, mastacprintflag, 0); %Distribution and
            Collections%                                                   0470
         %Now quit%                                                        0472
            jdquit();                                                      0473
         callsig(0, 0);%cause signal to reset prnary output and return%
                                                                           0474
         END.                                                              0475
      % Entry support-- expand Journal header%                            03481
        (setjheader)PROC(number, modstr, idflnm);                          03482
            %*setjheader*                                                  03483
               This procedure  basically converts an abbreviated Journal
               header (which is the result of Submit) into a full-fledged
               final format Journnal header.                               03484
               Along tne way, it:                                          03485
                  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 tne
                  names of all individuals/groups into the proper place in
                  the header (Specifically before the '/ preceding the
                  identlist)                                               03486
                  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.                                             03487
                  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 prccedure will put them unnder a branch
                  in the Journal catalog, and sometime in the future the
                  whollе mess willl b submitted to the catalog system for
                  catalog citation maintenance.                            03488
            %                                                              03489
         LOCAL TEXT POINTER idptr, z1, z2;                                 03490
         LOCAL idfnum, stid;                                               03491
         LOCAL STRING adsubcols[100], oldsubcols[300], tempsr[30];         03492
         REF number, modstr;                                               03493
         %Number is Journal Number, modstr will be used for returning any
         catalog manipulation commands for other entries%                  03494
         %This routine fills out the journal header%                       03495
         IF idflnm = 0 THEN                                                03497
            BEGIN                                                          03498
            idfnum ← 0;                                                    03499
            ON SIGNAL ELSE sigclose(idfnum := 0);                          03500
            idfnum ← open(0, jflname($"identfile"));                       03502
```

```
      END                                                          03504
   ELSE idfnum ← idflnm;                                           03505
   %Now fill out name fields%                                      03506
      %author field%                                               03507
         *lit* ← NULL;                                             03508
         IF NOT (FIND jwpl > ["Author(s): "] [1//] ↑idptr) THEN    03509
            snkerr($"Bad header statement");                       04165
         intids(0);                                                03510
         LOOP                                                      03511
            BEGIN                                                  03512
            IF NOT getids($idptr, $lit, 1, idfnum) THEN EXIT; %reads
            idents from jwpl (up to ';) and puts info into $lit
            (appended)...returns FALSE if no idents left%          03513
            *lit* ←*lit*,", ";                                     03514
            END;                                                   03515
         IF lit.L > 0 THEN lit.L ← lit.L-2; %ignore last comma%    03516
         CCPOS SF(jwpl);                                           03517
         FIND ["Author(s): "] ↑jwpl;                               03518
         ST jwpl ← SF(jwpl) jwpl, *lit*, jwpl SE(jwpl);            03519
   IF NOT idflnm THEN close(idfnum := 0);                          03520
   % Now fix up Sub-collection Field to reflect RFC, Distribution
   Groups%                                                         03521
      IF FIND SF(jwpl) (["RFC# "] 3$D) THEN rfcflg ← 2;            03522
      IF FIND SF(jwpl) ↑jwpl SF(*[galjsubcol($xlit,idfnum)]*) ↑zl
      THEN                                                         03523
         BEGIN                                                     03524
         *oldsubcols* ← zl SE(zl);                                 03525
         setjsubcol($oldsubcols);                                  03526
         END;                                                      03527
   %Now build any catalog modification commands dictated by
   parameter fields in the header%                                 03528
      IF &modstr THEN                                              03529
         BEGIN                                                     03530
         %Obsoletes%                                               03531
            getjobsoletes($xlit); % Nulls out xlit if none. %      03532
            IF xlit.L THEN                                         04172
               *modstr* ← "Obsoletes Document(s): ", *xlit*, EOL
                                                                   04173
            ELSE *modstr* ← NULL;                                  04174
         %Updates%                                                 03533
            getjupdates($xlit); % Nulls out xlit if none. %        03534
            IF xlit.L THEN                                         04175
               *modstr* ← *modstr*, "Updates Document(s): ", *xlit*;
                                                                   04176
         %Check to see if entries are not NULL%                    03535
            IF modstr.L THEN                                       04178
               *modstr* ← *number*, SP, *modstr*;                  04179
         END;                                                      03538
   RETURN END.
                                                                   03540
% Build J-catalog entries %                                       03542
   (updjcat)PROC(number, numused, linkstr, jcstid);                03543
      %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 the journal number of the document just
      entered%                                                     03544
```

```
   %jcstid is assumed to be the stid of the active file statement%
                                                                        03545
                                                                        03546
   LOCAL STRING tempsr[10];                                             03547
   LOCAL TEXT POINTER z1, z2;                                           03548
   REF number, linkstr;                                                 03549
   %ton 7%
   IF (jcstid ← namelook(jcstid, $"ACTIVE")) = endfil THEN
   snkerr($"Bad Jcat File");                                            03550
   IF NOT FIND SF(jcstid) ["Used = "] $NP ↑z1 1$D ↑z2 THEN               03551
      snkerr($"Bad Jcat File");                                         03552
   *tempsr* ← STRING(numused); %Get around L10 compiler glitch%         03553
   ST z1 ← SF(z1) z1, *tempsr*, z2 SE(z2), " ", *number*;               03554
   %Now save off name for distribution%                                 03555
      IF NOT FIND SF(jcstid) ("") ("] ↑z1 ['])] ↑z2 ←z2 THEN            03556
         snkerr($"Bad Jcat File");                                      03557
      *linkstr* ← z1 z2;                                                03558
   %toff 7%                                                             03559
   RETURN;                                                              03560
   END.
                                                                        03561
(jcatentry)PROC(astr1, astr2, astr3, hcflag, jcstid, headstr);          03562
                                                                        03563
   LOCAL stid, xstid;                                                   03564
   %make an entry in the catalog part of the jcat file.
      astr1 is user name, astr2 is file name for file, and astr3 is
      branch name%                                                      03565
   LOCAL TEXT POINTER z1, z2;                                           03566
   REF astr1, astr2, astr3, headstr;                                    03567
   %ton 8%                                                              03568
   stid ← jcstid;                                                       03569
   IF (xstid ← namelook(stid, $"JCATALOG"))= endfil THEN
   snkerr($"Bad Jcat File");                                            03570
   xstid ← ccopsta(xstid, levdown, jworkstid, FALSE, 0);                03571
   FIND SF(xstid) ↑z1 [',] [';] ↑z2;                                    03572
   ST z1 z2 ← *headstr*;                                                03573
   stid ← xstid;                                                        03574
   xstid ← cis(xstid, &astr1, $downstr);                                03575
   IF hcflag THEN                                                       03576
      BEGIN                                                             03577
      CCPOS SF(stid);                                                   03578
      FIND ["Location: "] ↑ z1 [';] ↑z2;                                03579
      ST xstid ← "Location of Document: ",  z1 z2;                      03580
      %toff 8%                                                          03581
      END                                                               03582
   ELSE                                                                 03583
      ST xstid ← "Link to document: (", *astr1*, ', , *astr2*, ', ,
      *astr3*, ":w) ";                                                  03584
   %toff 8%                                                             03585
   RETURN END.                                                          03586
                                                                        03587
(jdocfile)PROC(jcstid, fname, jwstid,  dirnam);
   %Given the stid of jcat, looks for a directory name statement,
   and checks that there is room for the indicated file in the
   directory.                                                           03588
      If everything is ok, it updates the file passed as jwstid into
      a new file in the new directory named fname.                      03589
      If directory is full, it looks for an ndirectory statement,
```

```
      and activates it, returning as above.                          03590
      It returns the name of the directory which was used            03591
      It signals if something goes wrong%                            03592
   LOCAL jdocstid;                                                   03593
   LOCAL TEXT POINTER z1, z2, z3, z4;                                03594
   LOCAL STRING tfname[50];                                          03595
   REF fname, dirnam, jwstid;                                        03596
   %First look for directory statment in JCAT%                       03597
      IF (jcstid ← namelook(jcstid, $"DIRECTORY")) = endfil THEN
      snkerr($"Directory list exhausted");                           03598
      IF NOT FIND SF(jcstid) ["Name: "] $NP ↑z1 [';] < CH ↑z2 > THEN
      snkerr($"Bad Jcat File");                                      03599
      *dirnam* ← z1 z2;                                              03600
   %Now open file%                                                   03601
      ON SIGNAL ELSE                                                 03602
         BEGIN                                                       03603
         %Delete current directory name, and look for ndirectory
         statements%                                                 03604
            FIND SF(jcstid) ['('] $NP ↑z1;                           03605
            ST z1 z1 ← 'F;                                           03606
            IF (jcstid ← namelook(jcstid, $"NDIRECTORY")) = endfil
            THEN                                                     03607
               SIGNAL(-1, $"Directory list exhausted");              03608
            FIND SF(jcstid) ['('] $NP ↑z1 CH ↑z2;                    03609
            ST z1 z2 ← NULL;                                         03610
         RETURN(jdocfile(jcstid, &fname, &jwstid, &dirnam));         03611
         END;                                                        03612
      *tfname* ← '<, *dirnam*, '>, *fname*, ".NLS", EOL;             03613
      IF NOT updtfil(jwstid.stfile, 1, $tfname) THEN snkerr($"Fatal
      Journal File Error");                                          03614
      setaccess(jwstid.stfile, jrnlaccess); %no toucheeee%           03615
      RETURN(jdocstid);                                              03616
   END.
                                                                     03617
% Hard copy delivery utility procedures %                           04145
   (inithcd)PROC; % Initialize for hard copy delivery. %              0476
      LOCAL TEXT POINTER z1, z2;                                      0477
      %Initialise Parameters for Hard Copy Distribution%              0478
         %Global stids and  file variables%                           0479
            docjfn ← hdrjfn ← docstrt ← lastfnum ← lptused ← lptjfn ←
            idfile ← prntfg ← 0;                                     0480
         %set up subsystem name for statistics%                      0481
            r1 ← oljsbn ← getsbn($"HCJDEL");                         0482
            !JSYS setnm;                                             0483
      %Now see which output processor and printer file we are to use%
                                                                     0484
         IF jdebug THEN                                              0485
            BEGIN                                                    0486
            %Debugging--use normal output processor and print file
            xlpt%                                                    0487
            *ptstr* ← "XLPT.TXT";                                    0488
            *oprocn* ← *opname*;                                     0489
            END                                                      0490
         ELSE                                                        0491
            BEGIN % real system..normal output processor and lpt%    0492
            *ptstr* ← NULL;                                          0493
```

```
        !JSYS gtad;                                                    0494
        dtfrmt(r1, $ptstr);                                            0495
        IF FIND SF(*ptstr*) [SP] ↑z1 ←z1 SE(z1) ↑z2 THEN ST z1 z2 ←
        NULL;                                                          0496
        *ptstr* ← "<DOCUMENTATION>", *ptstr*, "LPT.TXT";               0497
        *oprocn* ← *opname*;                                           0498
        END;                                                           0499
   %Now update ncdistfile from distfile%                               0500
      IF updhcdist() = 0 THEN                                          0501
        BEGIN                                                          0502
        typeas( $"Distribution File Empty");                           0503
        jdquit();                                                      0504
        SIGNAL( 0, $"Distribution file empty");                        0505
        END;                                                           0506
   RETURN;                                                             050
   END.
                                                                       0508
(printall)PROC(distribution, collections, nic);                        0509
   %This procedure prints all of the indicated copies of all
   documents in the ncdistfile%                                        0510
   LOCAL ncdstid, stid;                                                0511
   ncdstid ← 0;                                                        0512
   ON SIGNAL ELSE                                                      0513
      BEGIN                                                            0514
      sigclose(ncdstid.stfile := 0);                                   0515
      lptclose();                                                      0516
      END;                                                             0517
   ncdstid ← openlock(0, jfilname($"hcdistfile"));                     0518
   pfilnum ← 1;                                                        0519
   stid ← getsub(ncdstid);                                             0520
   WHILE stid # ncdstid DO                                             0521
      BEGIN                                                            0522
      printdoc(stid:= getsuc(stid), distribution, collections, nic);
                                                                       0523
      BUMP pfilnum;                                                    0524
      END;                                                             0525
   close(ncdstid.stfile :=0);                                         04268
   lptclose();                                                         0527
   RETURN;                                                             0528
   END.
                                                                       0529
(printdoc)PROC(ncdstid, distribution, collections, nic);               0546
   %Print out the various flavors of the document identified by
   distfile entry ncdstid.                                             0547
      Flags indicate which copies to print%                            0548
   LOCAL dwstid, stid, wrkstr;                                         0549
   LOCAL TEXT POINTER z1, z2;                                          0550
   LOCAL STRING jnum[5];                                               0551
   REF wrkstr;                                                        02934
   dwstid ← 0;                                                         0552
   prntfg ← TRUE;                                                      0553
   % allocate work string. %                                          02935
      &wrkstr ← getstring(2000, $dspblk);                             02936
   ON SIGNAL ELSE                                                      0554
      BEGIN                                                            0555
      close(dwstid.stfile := 0);                                       0556
```

```
    IF &wrkstr THEN freestring( &wrkstr:=0, $dspblk);                    02937
    CASE sysgnl OF                                                       02921
       = -4: NULL;                                                       02928
       ENDCASE                                                           02929
          BEGIN %Not called to rubout%                                   02930
          dismes(1, sysmsg);                                             02931
          IF sysgnl # sigdsf THEN RETURN;                                02932
          END;                                                           02933
    END;                                                                  0562
%get document number into jnum, and type it to the operator%              0563
    FIND SF(hcdstid) ↑z1;                                                 0564
    getjnm($z1, $jnum);                                                   0565
    crlf();                                                               0566
    typeas($"Document #");                                                0567
    typeas($jnum);                                                        0568
    crlf();                                                               0569
%set up document for printing%                                            0570
    %Get author names and addresses for address page%                     0571
       getdauth(hcdstid, &wrkstr);                                        0572
    %Set up te dwork file%                                                0573
       dwstid ← getdwork(hcdstid, &wrkstr);                               0574
%Do distribution copies first%                                            0575
    IF distribution THEN                                                  0576
       BEGIN                                                              0577
       stid ← getsub(hcdstid);                                            0578
       WHILE stid # hcdstid DO                                            0579
          pdistcopy(dwstid, stid := getsuc(stid), $jnum, &wrkstr);
                                                                          0580
       END;                                                               0581
% deallocate work string. %                                              02938
    freestring( &wrkstr:=0, $dspblk);                                     02939
%do system copies of public documemts%                                    04152
    IF NOT getcacc (hcdstid, 0, 0, 0) THEN                                04153
       BEGIN                                                              04154
       %Do collection copies%                                             0582
          IF collections AND NOT FIND SF(hcdstid) /EOL "Secondary
          Distribution Copy"/ THEN                                        0583
             BEGIN                                                        0584
             collcopy(dwstid, hcdstid, $"Master", $jnum);                 0585
             collcopy(dwstid, hcdsxid, $"Access", $jnum);                 0586
             END;                                                         0587
       %Do NIC copies%                                                    0588
          IF nic AND FIND SF(hcdstid) /"Sub-Collections: "/ ↑z1
          /';/ ↑z2 BETWEEN z1 z2(/"NIC"/) THEN                            0589
             BEGIN                                                        0590
             collcopy(dwstid, hcdstid, $"NIC", $jnum);                    0591
             nicsitecopies(dwstid, hcdstid);                              0592
             END;                                                         0593
       END;                                                               04155
    close(dwstid.stfile := 0);                                            0594
    RETURN;                                                               0595
    END.
                                                                          0596
(printnum)PROC(numstr, distribution, collections, nic);                   0530
    %Print the document indicated by numstr if there is an entry for
    it in the hcdistfile%                                                 0531
```

```
    LOCAL hcdstid, stid;                                                0532
    LOCAL STRING tempsr[10];                                            0533
    REF numstr;                                                         0534
    hcdstid ← 0;                                                        0535
    ON SIGNAL ELSE sigclose(hcdstid.stfile := 0);                       0536
    *tempsr* ← 'J, *numstr*;                                            0537
    hcdstid ← openlock(0, jflname($"hcdistfile"));                      0538
    IF (stid  ← namelook(hcdstid, $tempsr))= endfil THEN err($"No
    =such document in distfile");                                       0539
    pfilnum ← 1;                                                        0540
    printdoc(stid, distribution, collections, nic);                     0541
    lptclose();                                                         0542
    close(hcdstid.stfile := 0);                                         0543
    RETURN;                                                             0544
    END.                                                                0545

(pdistcopy)PROC(dwstid, adrstid, jnumber, wrkstr);                      02603
    %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%                                             02604
    LOCAL STRING adrstr[700];                                           02605
    LOCAL TEXT POINTER z1, z2, z3, z4;                                  02606
    REF jnumber, wrkstr;                                                02607
    IF discfull(300) THEN SIGNAL(sigdsf, $"*** TERMINATED: DISC FULL
    ***"); %check disc, also wait if load average high%                 02608
    COPOS SF(adrstid);                                                  02609
    xtrnam($adrstr, $swork, '(, ') );                                   02610
    ckident($adrstr, &wrkstr, 0);                                       02611
    IF phcopy(adrstid) THEN %A copy should be printed for this user%    02612
                                                                        02613
        BEGIN                                                           02614
        %Get his name and address%                                      02615
            getinam(&wrkstr, 0, $z1, $z2);                              02616
            laddress(&wrkstr, $adrstr, 0, 0, 0);
            *adrstr* ← z1 z2, EOL, *adrstr*, ".LMS=60;.GCR=5;",         02617
            *jnumber*, ".LMS=0;.GCR;";                                  02618
            %Now look for possible comments%
                IF FIND SF(adrstid) ([*"Note: "] ↑z1 [";;"] ↑z2  %New
                                                                        02619
                format%
                    /[".GCR=5;.GCR=5;"] ↑z1 [EOL EOL] ↑z2) ←z2 %old
                                                                        02620
                    Format%
                    THEN *adrstr* ← *adrstr*, ".GCR=4;.GCR=5;", z1 z2,  02621
                    EOL;                                                02622
        %Now print it%                                                  02623
            printit(dwstid, $adrstr, pfilnum);                          02624
        %Now tag statement as having been printed%
            IF FIND SE(adrstid) < ['*] > [" do-hc"] ↑z2 < [SP] ↑z1 THEN
                                                                        02625
            ST z1 z2 ← " hc-done-?\, *opstr*;                           02626
        END;                                                            02627
    RETURN;
    END.                                                                02628

(collcopy)PROC(dwstid, hcdstid, astr, jnum);                            0716
    %Print a 'Collection Copy' For collection identified by astr if
    it has not already been done.  Tag statement hcdstid accordingly%   0717
```

```
   LOCAL STRING tempsr[100];                                          0718
   LOCAL distexf, distid;                                             01170
   LOCAL TEXT POINTER pl;                                             01206
   REF jnum, astr;                                                    0719
   IF discfull(300) THEN SIGNAL(sigdsf, $"*** TERMINATED: DISC FULL
   ***"): %check disc, also wait if load average nigh%                01966
   *tempsr* ← *astr*, " Copy Printed";                                0720
   IF NOT FIND SF(ncdstid) [*tempsr*] THEN %It has not been printed%
                                                                      0721
      BEGIN                                                           0722
      IF FIND SF(*astr*) "Master" SF(dwstid) ["Distribution:"] ['/]
      ↑pl THEN %put out a couple of copies of the expanded
      distribution%                                                   01175
         BEGIN                                                        01177
         *tempsr* ← *jnum*, " Distribution";                          01171
         distid ← cis(dwstid, $tempsr, $downstr);                     01172
         distexf ← TRUE;                                              01173
         expdist(distid, $pl);                                        01174
         poutput (distid, $hdrjfn, $hdrwfn);                          01181
         chklpt();                                                    01182
         copylpt(FALSE, hdrjfn, 0, -1);                               01183
         copylpt(FALSE, hdrjfn, 0, -1);                               01184
         cdelgro(distid, distid, FALSE, 0); %delete distribution
         branch%                                                      01185
         END;                                                         01178
      *tempsr* ← *astr*, " Copy.LMS=60;.GCR=5;", *jnum*,
      ".LMS=0;.GCR;";                                                 0723
      printit(dwstid, $tempsr, pfilnum);                              0724
      *tempsr* ←  *astr*, " Copy";                                    0725
      tagjdel(ncdstid, $tempsr);                                      0726
      END;                                                            0727
   RETURN;                                                            0728
   END.
                                                                      0729
(printit)PROCEDURE(stid, astr, fnum);                                 0840
   LOCAL cntr, newdoc;                                                0841
   LOCAL TEXT POINTER z1, z2, z3;                                     0842
   REF astr;                                                          0843
   %Replace current name statement if astr # 0%                       0844
      IF &astr # 0 THEN                                               0845
         BEGIN                                                        0846
         CCPOS SF(stid);                                              0847
         IF NOT FIND [".LMS=24;
         "] ↑z1 ["
         .PEL;"] [EOL] ↑ z3 < [EOL]  [EOL] ↑z2 >  THEN                0848
            SIGNAL(2, $"Illegal Header in Printit");                  0849
         ST stid ← SF(stid) z1,*astr*,z2 z3, ".PSW=0;.HLT;", z3
         SE(stid);                                                    0850
         crlf();                                                      0851
         typeas(&astr); %let operator see what is happenning%         0852
         END                                                          0853
      ELSE                                                            0854
         BEGIN %Garden variety document%                              0855
         lastinum ← 0;                                                0856
         poutput(stid, $docjfn, $docwfn);                             0857
         chklpt();                                                    0858
```

```
        copylpt(TRUE, docjfn, 0, -1);                                   0859
        RETURN                                                          0861
        END;
                                                                        0862
  %output header%                                                       0863
     poutput(stid, $hdrjfn, $hdrwfn);                                   0864
  %If new file, get rest of document%                                   0865
     IF (lastfnum := fnum) # fnum THEN                                  0866
        BEGIN %New document%                                            0867
        IF NOT FIND SF(stid) ["PSW=0;.HLT;"] ↑z2 < ["P."] ↑z1 >
        THEN                                                            0868
           SIGNAL(2, $"Bad print file");                                0869
        SI z1 z2 ← NULL; %delete IRS and HALT%                          0870
        %output entire file%                                            0871
           poutput(stid, $docjfn, $docwfn);                             0872
        %Now get start of document from header file%                    0873
           r1 ← hdrjfn;                                                 0874
           IF NOT SKIP !JSYS sizef THEN                                 0875
              err(@"Print File Error");                                 0876
           docstrt ← r2-1;                                              0877
        END;                                                            0878
  %Now copy to lpt:%                                                    0879
     chklpt();                                                          0880
     copylpt(TRUE, hdrjfn, 0, -1); %header%                             ¤
     copylpt(FALSE, docjfn, docstrt, -1); %document%                    0882
  RETURN;                                                               0884
  END.
                                                                        0885
(poutput)PROC(stid, jfn, jfnnam);                                       0886
  LOCAL opjfn;                                                          0887
  REF jfn, jfnnam;                                                      0888
  %Close file if it is open%                                            0889
     IF jfn # 0 THEN  sysclose(jfn := 0);                               0890
  %now call processor%                                                  0891
     tda.dacsp ← stid;                                                  0892
     coutproc (&jfnnam, &tda, cpprdv, 0 %flags for output processor
     %);                                                                0893
  %Now re-open file%                                                    0894
     jfn ← sgtjfn(getgtjflg(write, 0, oldvrsn), &jfnnam, $lit);         0895
     IF NOT sysopen(jfn, readwrite, chrtyp, $lit) THEN                  0896
        BEGIN                                                           0897
        %May have failed for timing reasons%                            0898
        r1 ← 1000;                                                      0899
        !JSYS dismsc;                                                   0900
        IF NOT sysopen(jfn, readwrite, chrtyp, $lit) THEN               0901
           err($lit);                                                   0902
        END;                                                            0903
  RETURN END.
                                                                        0904
(nicsitecopies)PROC(dwstid, hcdstid);                                   0739
  RETURN;                                                               0740
  END.
                                                                        0741
(getdauth)PROC(stid, authsr);                                           02569
  %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%                                                    02570
   LOCAL idfno;                                                         02571
   LOCAL STRING tempsr[30], tempadsr[150], tempauthor[50];              02572
   LOCAL TEXT POINTER z1, z2;                                           02573
   REF authsr;                                                          02574
   idfno ← 0;                                                           02575
   ON SIGNAL ELSE                                                       02576
      BEGIN                                                             02577
      sigclose(idfno := 0);                                             02578
      IF sysgn1 # -4 THEN SIGNAL(1);                                    02579
      END;                                                              02580
   IF NOT FIND SF(stid) ["Author(s): "] ['/] $NP ↑z1 ('&/'↑/) ↑z2
   THEN                                                                 02581
      BEGIN                                                             02582
      IF NOT FIND SF(stid) [".HJOURNAL="] [NP 1$2D '- 3L] < [1$NP]
      ↑z2 ['" / ';] > CH ↑z1 THEN SIGNAL(1, $"Bad Distfile Entry");
                                                                        02583
      *tempauthor* ← z1 z2, '; ';                                       02584
      FIND SF(*tempauthor*) ↑z1 ('&/'↑/) ↑z2;                           02585
      END;                                                              02586
   *authsr* ← NULL;                                                     02587
   idfno ← open(0, jflname($"identfile"));                              02588
   %Get the names of the authors%                                       02589
      intids(0);                                                        02590
      WHILE getids($z1, &authsr, 1, idfno) DO                           02591
         *authsr* ← *authsr*, EOL;                                      02592
   %Now get the addresss of the frst author%                            02593
      IF (authsr.L = 0) OR NOT FIND z2 L $(LD / '- / '') ↑z1 THEN
      err($"Illegal Author Field");                                     02594
      *tempsr* ← z2 z1;                                                 02595
      IF NOT ckident($tempsr, $xlit, idfno) THEN err($"Illegal
      Author");                                                         02596
   %Now extract address and put it all together%                        02597
      laddress($xlit, $tempadsr, 0, 0, idfno);                          02598
      close(idfno := 0);                                                02599
      *authsr* ← *authsr*, *tempadsr*;                                  02600
   RETURN;                                                              02601
   END.
                                                                        02602
(getdwork)PROC(ncdstid, authsr);                                         0627
   %Open and set up dwork file for printing.                             0628
      Return stid of dwork file if ok, zero if not.                      0629
      %                                                                  0630
   REF authsr;                                                           0631
   LOCAL dwstid, docstid;                                                0632
   LOCAL TEXT POINTER z1, z2, z3, t1;                                    0633
   dwstid ← docstid ← 0;                                                 0634
   ON SIGNAL ELSE                                                        0635
      BEGIN                                                              0636
      close(dwstid.stfile := 0);                                         0637
      close(docstid.stfile := 0);                                        0638
      END;                                                               0639
   %Open dwork, and set up the origin statement%                         0640
      dwstid ← openlock(0, jflname($"distwork"));                        0641
      enablaccess(dwstid.stfile, jrnlaccess);                            0642
      resetf(dwstid.stfile); % reset file %                              0643
```

```
    FIND SF(hcdstid) [')] ↑z1 ( $NP '( [')] / ) ↑z2; %delimit
    link%                                                          0644
    %TEMPORARY kludge for output processor glitch when no title
    (.H1) directive%                                               02211
       IF (FIND z2 [".HJOURNAL="] CH ['" ';] $NP ↑t1) AND NOT
       (FIND "Title:") THEN ST t1 t1 ← "Title: .H1=", '", ' , '",
       "; ";                                                       02210
    FIND SE(z2) (([EOL EOL] SEOL > 2EOL) / ) > ↑z3; %position z3
    at end, but before "Master Copy Printed", etc.%                02209
    FIND z3 > (".PEL;" (([EOL ("Master" / "Access" / "Engelbart")]
    < [EOL] >) / SE(z3)) ↑z3); %move past comment if here%         02213
    ST dwstid ← *authsr*, ".LMS=20;
    .GCR=3; To: .LMS=24;

    .PEL;
    .LMS = 0;", SF(hcdstid) z1, z2 z3;                             0648
  %Now set up the rest of the file%                                0649
    IF docstid ← jdelloaddoc(hcdstid) THEN                         0650
       BEGIN                                                       0651
       ccopgro(dwstid, levsuc, docstid ← getsub(docstid),
       getail(docstid), FALSE, 0);                                 0652
       close(docstid.stfile := 0);                                 0653
       END;                                                        0654
  RETURN(dwstid);                                                  0655
  END.                                                             0656

(getjnm)PROC(ptr, numsr);                                          04225
  %Get journal number field from distfile entry %                  04226
  LOCAL TEXT POINTER z1, z2;                                       04259
  REF ptr, numsr;                                                  04229
  IF NOT (FIND ptr > ['(] ('J/'j) ↑z1 4$5D ↑z2 ) THEN              04256
     err($"Bad distfile header entry");                            04257
  *numsr* ← z1 z2;                                                 04258
  RETURN;                                                          04254
  END.                                                             04255

(jdelloaddoc)PROC(stid);                                           0657
  %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%                                      0658
  LOCAL docstid, adstr[40];                                        0659
  LOCAL TEXT POINTER z1;                                           0660
  LOCAL STRING user[20], filename[50], statname[30], vsp[15];      0661
  docstid ← 0;                                                     0662
  ON SIGNAL ELSE                                                   0663
     BEGIN                                                         0664
     close(docstid.stfile := 0);                                   0665
     SIGNAL(1);                                                    0666
     END;                                                          0667
  IF FIND SF(stid) [')] ↑z1  ["Hard Copy--Location"] THEN
  RETURN(0);                                                       0668
  %Now parse link---z1 points to spaces before it%                 0669
     lnkpspc(1, $z1, $user, $filename, $statname, $vsp, $adstr);
                                                                   0670
     *filename* ← *filename*, ".NLS";                              0671
  %Now open file and find proper branch%                           0672
```

```
      docstid.stpsid ← origin;                                          0673
      docstid.stfile ← open(0, $filename);                              0674
      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 %                                                 0675
      makeptr(docstid, $z1);                                            0676
      specreg($statname, nametyp, $z1);                                 0677
      IF z1 = endfile THEN                                              0678
         BEGIN                                                          0679
         *lit* ← *filename*, " Branch ", *statname*, " Not Found";
                                                                        0680
         err($lit);                                                     0681
         END;                                                           0682
      RETURN(z1);                                                       0683
   END.
                                                                        0684
(hcstatus)PROC;                                                         0787
   LOCAL hcdstid, stid1, stid2, distcount, collcount, niccount,
   count, idjfn;                                                        0788
   LOCAL TEXT POINTER z1, z2;                                           0789
   %return number of expedited printouts to be done, no. of normal
   docs%                                                                0790
   hcdstid ← 0;                                                         0791
   ON SIGNAL ELSE                                                       0792
      BEGIN                                                             0793
      sigclose(hcdstid.stfile := 0);                                    0794
      sigclose(idjfn := 0);                                             0795
      END;                                                              0796
   hcdstid.stpsid ← origin;                                             0797
   hcdstid.stfile ← open(0, jflname($"hcdistfile"));                    0798
   idjfn ← open(0, jflname($"Identfile"));                              0799
   stid1 ← getsub(hcdstid);                                             0800
   distcount ← collcount ← niccount ← 0;                                0801
   WHILE stid1 # hcdstid DO                                             0802
      BEGIN                                                             0803
      stid2 ←  getsub(stid1);                                           0804
      WHILE stid2 # stid1 DO                                            0805
         BEGIN                                                          0806
         distcount ← distcount + phcopy(stid2);                         0807
         stid2 ← getsuc(stid2);                                         0808
         END;                                                           0809
      collcount ← collcount +                                           0810
         (NOT FIND SF(stid1) ["Master Copy Printed"]) +                 0811
         (NOT FIND SF(stid1) ["Access Copy Printed"]) +                 0812
         (NOT FIND SF(stid1) ["Engelbart Copy Printed"]);               0813
      niccount ← niccount + ((FIND SF(stid1) ["Sub-Collections: "]
      ↑z1 [';] ↑z2 BETWEEN z1 z2(["NIC"])) AND NOT FIND SF(stid1)
      ["NIC Copy Printed"]);                                            0814
      stid1 ← getsuc(stid1);                                            0815
      END;                                                              0816
   close(idjfn := 0);                                                   0817
   close(hcdstid.stfile := 0);                                          0818
   RETURN(distcount, collcount, niccount);                              0819
   END.
                                                                        0820
(jdquit)PROCEDURE;                                                      0821
```

```
   %Print out hcdistfile if printing has been done, and garbage
   collect distribution file%                                          0822
   LOCAL hcdstid, svspc, svstd;                                        0823
   hcdstid ← 0;                                                        0824
   ON SIGNAL ELSE sigclose(hcdstid.stfile := 0);                       0825
   IF prntfg  THEN                                                     0826
      BEGIN                                                            0827
      prntfg ← FALSE;                                                  0828
      % print out distfile%                                           0829
         % save viewspecs and stid fr they will be changed by outptr
                                                                       04263
         %                                                            04261
            svspc ← tda.davspec;                                       04262
            svstd ← tda.dacsp;
         outptr(IF jddebug THEN $"<Duval1>hcdistfile" ELSE
         $"<Journal>hcdistfile", &tda, 1, TRUE);                       0830
         % restore dacsp and davspec %                                04264
            tda.davspec ← svspc;                                       04265
            tda.dacsp ← svstd;                                         04266
      %garbage collect hcdistfile%                                     0831
         hcdstid ← openlock(0, jfiname($"hcdistfile"));                04267
         gdistf(hcdstid:=0);                                           0833
       END;                                                            0834
   RETURN;                                                             0838
   END.                                                                0839

(lptclose)PROC;                                                        0730
   %Close out all of the lpt machinery%                                0731
   IF lptjfn THEN sysclose(lptjfn := 0);                               0732
   IF docjfn THEN sysclose(docjfn := 0);                               0733
   IF hdrjfn THEN sysclose(hdrjfn := 0);                               0734
   %Global stids and  file variables%                                  0735
      docstrt ← lastfnum ← lptused ← idfile ← 0;                       0736
   RETURN;                                                             0737
   END.                                                                0738

(copylpt)PROC(resflg, jfn, startb, stopb);                             0905
   LOCAL count;                                                        0906
   %First set file pointer to start%                                   0907
      r1 ← jfn; r2 ← startb; IF NOT SKIP !JSYS sfptr THEN err($"I/O
      JSYS error");                                                    0908
      %Now set eof%                                                    0909
         IF stopb = -1 THEN                                            0910
            BEGIN                                                      0911
            %get size of file%                                         0912
            r1 ← jfn; IF NOT SKIP !JSYS sizef THEN err($"I/O JSYS
            error");                                                   0913
            stopb ← r2;                                                0914
            END                                                        0915
         ELSE                                                          0916
            BEGIN                                                      0917
            %set size of file%                                         0918
            r1.RH ← jfn; r1.LH ← 12B; r2 ← -1; r3 ← stopb;             0919
            !JSYS chfdb;                                               0920
            END;                                                       0921
      %Send out restore if necessary%                                  0922
         %output processor paginates for us now%                       0923
```

```
        %Get count and output%                                                  0924
           count ← stopb - startb;                                              0925
           lptused ← lptused + count;                                          01416
           WHILE count > 0 DO                                                   0926
              BEGIN                                                             0927
              %get string from input%                                           0928
                 count ← count - 2560;                                          0929
                 r3 ← -(IF count < 0 THEN 2560+count ELSE 2560);                0930
                 r1 ← jfn; r2 ← chbptr(0) + $ckpag;                             0931
                 !JSYS sin; %read string from input file%                       0932
              %copy to lpt%                                                     0933
                 r1 ← lptjfn; r2 ← chbptr(0) + $ckpag;                          0934
                 r3 ← -(IF count < 0 THEN 2560+count ELSE 2560);                0935
                 !JSYS sout;                                                    0936
              END;                                                              0937
        RETURN                                                                  0938
        END.                                                                    0939

                                                                                0940
    (chklpt)PROC;                                                               0941
       LOCAL cntr;                                                              0942
       IF lptjfn # 0 THEN BEGIN                                                 0943
          IF lptused < 75000 THEN RETURN ELSE sysclose(lptjfn:=0);              0945
          END;                                                                  0946
       lptused ← 1;
       IF (lptjfn ← sgtjfn(getgtjflg(write, 0, 0), $ptstr, $lit)) = 0           0947
       THEN err($lit);                                                          0948
       FOR cntr ← 0 UP 1 UNTIL > 90 DO                                          0949
          IF sysopen(lptjfn, write, lpttype, $lit) THEN RETURN                  0950
          ELSE pause(20000);                                                    0951
       lptjfn ← lptused ← 0;                                                    0952
       err($"Printer busy too long");
       END.                                                                     0953

                                                                               02077
    (tagjdel)PROC(stid, astr);                                                 02078
       REF astr;
       %Tag the statement stid with string + "Printed By " OPERATOR"%           02079

       ST stid ← SF(stid) SE(stid), EOL, *astr*, " Printed by ",                02080
       *opstr*;                                                                 02081
       RETURN;
       END.                                                                     02082

                                                                                03728
% Build Distribution file enties %
    (dsecdoc) PROC(number, nethcflag, usetejlink, jcstid, idnum);
    %process secondary distribution request in jworkfile into distfile%         03770

                                                                                03771
       LOCAL jxcstid;                                                           03772
       LOCAL TEXT POINTER z1, z2, z3;                                           03773
       LOCAL STRING notstr[55];                                                 03774
       REF number;                                                              03775
       jxcstid ← 0;                                                             03776
       ON SIGNAL ELSE                                                           03777
          BEGIN                                                                 03778
          close(jxcstid.stfile :=0);                                            03779
          RETURN (FALSE);                                                       03780
          END;
```

```
   FIND jwp1 > ([")("(J" / "(j"] ↑z1) ←2z1 SE(jworkstid) > ↑z3;
   ST jworkstid ← z1 z3, EOL, "Secondary Distribution Copy";
   IF FIND jwp1 ["(Expedite)"] ↑z3 < ['(] ↑z2 >  THEN ST z2 z3 ←
   NULL;
   z2 ← getsub(jworkstid);
   z1 ← getsuc(z2);
   FIND jwp1 [')] ↑z3;
   IF NOT usetejlink THEN %must get link from catalog%
      BEGIN
      IF NOT (z2 ← gtcatent (&number, 0, jcstid.stfile, FALSE)) AND
      NOT (z2 ← jxcstid ← gtcatent (&number, 0, 0, FALSE)) THEN err
      ($"No such document");
      z2 ← getsub(z2);
      END;
   IF FIND SF(z2) ["Link"] ['(] ↑z2 ←z2 THEN
      BEGIN
      ST z3 z3 ← SP, z2 SE(z2); %link%
      END;
   %make notstr%
      z2 ← getsuc(z1);
      *notstr* ← "(Secondary Distribution Copy from ", SF(z2)
      SE(z2), ');
   %set up ident list pointer%
      FIND SF(z1) ↑z1;
   diststid ← ccopsta(diststid, levsuc, jworkstid, 0, 0);
   distdl(diststid, $z1, idnum, $notstr, nethcflag);
   close (jxcstid.stfile:=0);
   RETURN(TRUE); END.

(distdoc)PROC(astr, linkuser,  number, mesflg, headstr, idnum,
nethcflag, recordedflg);
   %distribute document identified by jworkstid%
   %do net and hc delivery if nethcflag%
   LOCAL stid;
   LOCAL TEXT POINTER z1, z2, z3, z4, z5;
   REF astr, number, linkuser, headstr;
   %astr contains file name upon call, Return link to document in
   astr%
   %ton 9%
   diststid ←  ccopsta(diststid, levsuc, jworkstid, FALSE, 0);
   FIND SF(diststid) ↑z1 [',] [';] ↑z2;
   ST z1 z2 ← *headstr*;
   CASE recordedflg OF
      = TRUE:
         BEGIN
         CASE mesflg OF
            =0:  *astr* ← '(, *linkuser*, ", ", *astr*,  ", 1:w)";

            =1: *astr* ← '(, *linkuser*, ", ", *astr*, %journal file
            name%
               ", J", *number*, ":gw) ";
            =hcopyv:  *astr* ← "(Hard Copy--",  *linkuser*, ", ",
            *astr*,  ", )";
            ENDCASE snkerr($"Illegal Message Type");
         END;
      ENDCASE
```

```
        BEGIN % unrecorded: resides in jworkstid.stfile %                    03828
        % make branch name start with "U" instead of "J" %                   03829
           IF FIND > z1 ["(J"] ↑z3 ↑z4 ←z4 THEN                              03830
              ST z1 ← z1 z4, 'U, z3 SE(z1);                                  03831
        % make link to actual location, a file with an "NLU"
        extension %                                                          03832
           FIND SF(*astr*) ↑z5;                                              03833
           FIND z5 ['<] ↑z5 [">] ↑z3 ↑z2 ←z2;                                03834
           *astr* ← '(, z5 z2, ', , z3 SE(*astr*),  ", :1 )";                03835
        END;                                                                 03836
     IF mesflg # hcopyv THEN                                                 03837
        BEGIN                                                                03838
        FIND SF(diststid) [')]  ↑z1;                                         03839
        ST z1 z1 ← "  ", *astr*; %Insert link into header for
        distribution%                                                        03840
        END;                                                                 03841
     %now insert individuals on distribution list%                           03842
        IF FIND SF(diststid)["Distribution: "] ['//] ↑ z1 THEN               03843
           distdl(diststid, $z1, idnum, IF mesflg = hcopyv THEN $"Hard
           Copy Document" ELSE 0, nethcflag);                                03844
     %now provide for author copies%                                         03845
        IF FIND SF(diststid) ["Author(s):"] ['//] ↑ z1 THEN                  03846
           distdl(diststid, $z1, idnum, $"Author Copy", nethcflag)
                                                                             03847
        ELSE snkerr($"System Error (No Author Field)");                      03848
     %toff 9%                                                                03849
     RETURN END.
                                                                             03850
(distdl)PROC(dstid, ptr, idnum, note, nethcflag);                            03851
     %This procedure does the actual manipulation of the distribution
     file to put entries in for the individual reciepients.                  03852
        dstid is te stid of the branch head in the distribution fule
                                                                             03853
        ptr is a pointer to the distribution list                            03854
        idnum is the identification file number (if non-zero)                03855
        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.                                   03856
     %                                                                       03857
     LOCAL idfnum, stid, ldelflags, netflag, hcflag;                         03858
     LOCAL TEXT POINTER z1, z2, z3;                                          03859
     LOCAL STRING ddstr[30];                                                 03860
     REF ptr, note;                                                          03861
     netflag ← nethcflag.delnet;                                              03862
     hcflag ← nethcflag.delhc;                                               03863
     IF idnum = 0 THEN                                                       03864
        BEGIN                                                                03865
        idfnum ← 0;                                                          03866
        ON SIGNAL ELSE sigclose(idfnum := 0);                                03867
        idfnum ← open(0, jflname($"identfile"));                             03868
        END                                                                  03869
     ELSE idfnum ← idnum;                                                    03870
     intids(0);                                                              03871
     LOOP                                                                    03872
        BEGIN                                                                03873
```

```
        *lit* ← NULL;                                                   03874
        IF NOT getids(&ptr, $lit, 0, idfnum) THEN EXIT;                 03875
        ldelflags ← ldelivery(asrref($lit));                            03876
        *ddstr* ← NULL;                                                 03877
        IF netflag AND ldelflags.delnet THEN                            03878
           *ddstr* ← *ddstr*, " do-net";                                04167
        IF hcflag AND ldelflags.delhc THEN                              03879
           *ddstr* ← *ddstr*, " do-hc";                                 04168
        IF ldelflags.delol THEN                                         03880
           *ddstr* ← *ddstr*, " do-ol";                                 04169
        IF ddstr.L = 0 THEN REPEAT LOOP;                                03881
        getiid($lit, 0, $z1, $z2);                                      03882
        *lit* ← '(, z1 z2, '); %ident%                                  03883
        IF gbotids($z3) THEN                                            03884
           FIND z3 ELSE FIND ptr;                                       04170
        IF FIND < CH > ') THEN % comment field present %                03885
           BEGIN                                                        03886
           IF NOT FIND < CH ↑z2  ['('] >  CH ↑z1 THEN                   03887
              snkerr($"System Error (Bad distribution List)");          03889
           % change do-ol by adding 'a (action) %                       03890
              IF ldelflags.delol THEN                                   03891
                 IF FIND " / ACTION /" THEN *ddstr* ← *ddstr*, 'a;
                                                                        03892
                 % format must aggree with that in (nls, csendmail,
                 convjdist) %                                           04171
           IF &note THEN                                                03893
              *lit* ← *lit*, EOL, "Note: ", z1 z2, EOL, *note*, ";;"
                                                                        03894
                                                                        03895
           ELSE                                                         03896
              *lit* ← *lit*, EOL, "Note: ", z1 z2, ";;";                03897
           END                                                          03898
        ELSE
           IF &note THEN *lit* ← *lit*, EOL, "Note: ", *note*, ";;";
                                                                        03899
        *lit* ← *lit*, " *DEL:", *ddstr*;                               03900
        cls(dstid, $lit, $downstr);                                     03901
        END;                                                            03902
     IF NOT idnum THEN close(idfnum);                                   03903
     RETURN END.
                                                                        03904
% Distribution file manipulation-- garbage collection, updating %       04147
  (gdistf)PROC(hcostid);                                                01044
     %Garbage collect hard copy distribution file%                      01045
     LOCAL stid, stid1, idjfn;                                          01046
     LOCAL TEXT POINTER z1,z2;                                          01047
     LOCAL STRING tempsr[50];                                           01048
     %Set up  subsy name%                                               01049
        r1 ← getsbn($"GCOLDF"); !JSYS setnm;                            01050
     idjfn ← 0;                                                         01051
     ON SIGNAL ELSE                                                     01052
        BEGIN                                                           01053
        sigclose(idjfn := 0);                                           01054
        sigclose(hcdstid.stfile := 0);                                  01055
        END;                                                            01056
     %open hcdist file%                                                 01057
        IF NOT hcdstid.stfile THEN hcdstid ← openlock(0,
```

```
      jflname($"hcdistfile"));                                          01058
      idjfn ← open(0, jflname($"identfile"));                           01059
   %loop through file looking for stuff to delete%                      01060
      stid ← getsub(hcdstid);                                           01061
      WHILE stid # hcdstid DO                                           01062
         BEGIN                                                          01063
         stid1 ← getsub(stid);                                          01064
         WHILE stid1 # stid DO                                          01065
            BEGIN                                                       01066
            CCPOS SF(stid1);                                            01067
            xtrnam($tempsr, $swork, -1);                                01068
            ckident($tempsr, $xlit, idjfn);                             01069
            IF NOT (phcopy(stid1) OR donline(stid1) OR
            dnetdel(stid1)) THEN                                        01070
               cdelsta(stid1 := getsuc(stid1), FALSE, 0)                01071
            ELSE stid1 ← getsuc(stid1);                                 01072
            END;                                                        01073
         IF getsub(stid) = stid THEN                                    01074
            BEGIN %all copies printed%                                  01075
            CCPOS SF(stid);                                             01076
            IF                                                          01077
               (                                                        01973
                  (FIND ↑z1 /"Master Copy Printed"/)                    01968
                  AND (FIND z1 /"Access Copy Printed"/)                 01078
                  %no NIC copies for now%                               01970
                     %AND NOT((FIND z1 /"Sub-Collections: "/ ↑z1
                     /';/ ↑z2 BETWEEN z1 z2 (/"NIC"/)) AND NOT FIND
                     SF(z1) /"NIC Copy Printed"/)%                      01971
               )                                                        01969
               OR (FIND z1 /EOL "Secondary Distribution Copy"/)         01972
               OR getcacc (stid, 0, 0, 0) %private document%            04151
               THEN cdelsta(stid := getsuc(stid), FALSE, 0)             01080
            ELSE stid ← getsuc(stid);                                   01081
            END                                                         01082
         ELSE stid ← getsuc(stid);                                      01083
         END;                                                           01084
   %Update and exit%                                                    01085
      closeu(hcdstid.stfile := 0);                                      01086
      close(idjfn);                                                     01087
      r1 ← getsbn($"HCJDEL");                                           01088
      !JSYS setnm;                                                      01089
      RETURN                                                            01090
      END.
                                                                        01091
(updhcdist)PROC;                                                         0742
   %Update DISTFILE from TDISTFILE, and return TRUE if there is
   anything in the distribution file%                                    0743
   LOCAL ldiststid, hcdstid, retval, stid, stidd, deldel;                0744
   LOCAL STRING adsr/45/;                                                0745
   %Initialise LOCALS%                                                   0746
      hcdstid ← 0;                                                       0747
      hcdstid.stpsid ← origin;                                           0748
      ldiststid ← hcdstid;                                               0749
   %Set Signal to close files%                                           0750
      ON SIGNAL ELSE                                                     0751
         BEGIN                                                           0753
```

```
        sigclose(hcdstid.stfile);                                          0754
        sigclose(ldiststid.stfile);                                        0755
        END;                                                               0756
    ldiststid ← openlock(0, jflname($"distfile"));                         0757
    IF (getsub(ldiststid)).stpsid = origin THEN                            0758
       BEGIN                                                               0759
       close(ldiststid.stfile);                                            0760
       RETURN(TRUE); %Assume that there will always be something in
       the dist file%                                                      0761
       END;                                                                0762
    hcdstid ← openlock(0,jflname($"hcdistfile"));                          0763
    enablaccess(hcdstid.stfile, jrnlaccess);                               0764
    enablaccess(ldiststid.stfile, jrnlaccess);                             0765
    cmovgro(stid←getail(getsub(hcdstid)), levsuc, ldiststid ←
    gethed(getsub(ldiststid)), getail(ldiststid), FALSE, 0);               0766
    closeu(ldiststid.stfile := 0);                                         0767
    UNTIL (stid ← getsuc(stid)).stpsid = origin DO                         0768
       BEGIN                                                               0769
       *xlit* ← SP;                                                        0770
       stidd ← getsub (stid);                                              0771
       UNTIL stidd = stid DO BEGIN                                         0772
          CCPOS SF(stidd);                                                 0773
          xtrnam ($adsr, $swork, -1);                                      0774
          IF adsr.L = 0 OR FIND SF(*xlit*) [SP *adsr* SP]                  0775
             THEN deldel ← stidd                                           0776
             ELSE BEGIN                                                    0777
                *xlit* ← *xlit*, *adsr*, SP;                               0779
                deldel ← (IF FIND [EOL] ("Note: *Exclude") THEN stidd
                ELSE FALSE);                                               01166
                END;                                                       0780
          stidd ← getsuc(stidd);                                           01167
          IF deldel THEN cdelgro(deldel, deldel, FALSE,0);                 01168
          END;                                                             0781
       END;                                                                0782
    retval ← (getsub(hcdstid)).stpsid # origin;                            0783
    closeu(hcdstid.stfile := 0); %update file%                             0784
    RETURN(retval);                                                        0785
    END.                                                                   0786

% Message files %                                                          03905
  (entrjdoc)PROC(stid, number, headstr);                                   03906
    %Enter journal documnnt into active file as identified by stid%
                                                                           03907
    LOCAL TEXT POINTER z1, z2;                                             03908
    REF number,headstr;                                                    03909
    %ton 5%                                                                03910
    %first copy over document%                                             03911
       stid ← getail(getsub(stid));                                        03912
       stid← ccopgro(stid, levsuc, jworkstid, jworkstid, FALSE, 0);
                                                                           03913
       FIND SF(stid) ↑z1 [',] [';] ↑z2;                                    03914
       ST z1 z2 ← *headstr*;                                               03915
       stid.stpsid ← origin;                                               03916
       ST SF(stid) ← SF(stid) SE(stid), " ", *number*; %update
       header%                                                             03917
    %toff 5%                                                               03918
```

```
   RETURN END.                                                          03919
(findactive)PROC(numsmt, jcstid);                                       03920
   %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.                               03921
   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 %                                                             03922
   LOCAL usedsmt, stid;                                                 03923
   LOCAL TEXT POINTER z1, z2;                                           03924
   LOCAL STRING tempsr[15];                                             03925
   %ton 4%                                                              03926
   %First look for active%                                              03927
      IF (stid ← namelook(jcstid,$"ACTIVE"))= endfil1 THEN
      RETURN( NEWACTIVE(jcstid), numsmt);%get new active file% %need
      new active file%                                                  03928
   %now check room here%                                                03929
      %first get current number of statements in file%                  03930
         FIND SF(stid) ("Used = "/ $NP ↑z1 $D ↑z2;                      03931
         *tempsr* ←  z1 z2; %number of used statements%                 03932
         usedsmt ← VALUE($tempsr);                                      03933
      %now check to see if room%                                       03934
         IF usedsmt+numsmt > jmaxsmt THEN                               03935
            BEGIN %no more room in this active%                         03936
            relactive(stid, jcstid); %release it%                       03937
            RETURN(newactive(jcstid), numsmt);                          03938
            END                                                         03939
         ELSE RETURN(openactive(stid, FALSE), usedsmt+numsmt);          03940
   END.                                                                 03941
(newactive)PROC(jcstid);                                                03942
   %This sets up a new active file, and opens it, returning the stid
   of the origin statement%                                             03943
   LOCAL STRING tempsr[35];                                             03944
   LOCAL TEXT POINTER tp1, tp2;                                         03945
   IF (jcstid := getail(getsub(jcstid))) = jcstid  THEN                 03946
      *tempsr* ←  "(ACTIVE) (JRNL1) Used = 0"                           03947
   ELSE                                                                 03948
      BEGIN                                                             03949
      COPOS SF(jcstid);                                                 03950
      IF NOT FIND $NP "(JRNL" ↑tp1 1$D ↑tp2 THEN                        03951
         SIGNAL(-5, $"Bad Entry in Jcat File"); %Make it look like a
         bad file%                                                      03952
      *tempsr* ← tp1 tp2;                                               03953
      bumpstr($tempsr);                                                 03954
      *tempsr* ← "(ACTIVE) (JRNL", *tempsr*, ") Used = 0";              03955
      END;                                                              03956
   jcstid ← cis(jcstid, $tempsr, $sucdir);                              03957
   RETURN(openactive(jcstid, TRUE));                                    03958
   END.                                                                 03959
(openactive)PROC(stid, newfile);                                        03960
   %This routine fetches the active name from the statement
   identified by stid, and opens the file, returning the stid of the
   origin statement%                                                    03961
```

```
    %Note that lock is not a concern because JCAT MUST BE OPEN which
    is locked%                                                          03962
    LOCAL STRING tempsr[15];                                            03963
    LOCAL TEXT POINTER tp1, tp2;                                        03964
    CCPOS SF(stid);                                                     03965
    IF NOT FIND $NP "(ACTIVE) (" ↑tp1 [')] ↑tp2 ←tp2 THEN               03966
       err($"Jctl Error");                                              03967
    *tempsr* ← tp1 tp2;                                                 03968
    IF newfile THEN                                                     03969
       BEGIN                                                            03970
       stid ← openwk(0, jflname($tempsr));                              03971
       setaccess(stid.stfile, jrnlaccess);                              03972
       closeu(stid.stfile);                                             03973
       END;                                                             03974
    stid ← 0;                                                           03975
    stid ← openlock(0, jflname($tempsr));                               03976
    %toff 4%                                                            03977
    RETURN(stid);                                                       03978
    END.                                                                03979

(relactive)PROC(stid);                                                  03980
    %This procedure releases the active identified in the jcat by
    stid%                                                               03981
    LOCAL TEXT POINTER tp1;                                             03982
    CCPOS SF(stid);                                                     03983
    FIND ["(ACTIVE) "] ↑tp1;                                            03984
    ST tp1 ← tp1 SE(tp1);                                               03985
    RETURN END.                                                         03986

% Read fields from distribution file entry. %                           04149
(phcopy)PROC(stid);                                                     01092
    %Return true if hard copy needs to be printed for statement
    identified by stid%                                                 02064
    IF FIND SE(stid)< ['*] > "*DEL:" [" do-hc"] THEN RETURN(TRUE)
    ELSE RETURN(FALSE);                                                 02065
    END.                                                                01108

(donline)PROC(stid);                                                    01109
    %Return (1) true if on line distribution should be done for
    statement identified by stid, and (2) true if ACTION type send.%
                                                                        01110
    LOCAL retflg;                                                       02568
    retflg ← FALSE;                                                     02545
    IF FIND SE(stid) ['*] > "*DEL:" [" do-ol"] THEN                     02066
       BEGIN                                                            02543
       IF FIND 'a THEN retflg ← TRUE;                                   02544
       RETURN(TRUE, retflg);                                            02542
       END                                                              02546
    ELSE RETURN(FALSE, FALSE);                                          02547
    END.                                                                01124

(dnetdel)PROC(stid);                                                    02067
    %Return true if net distribution should be done for statement
    identified by stid.%                                                02068
    IF FIND SE(stid)< ['*] > "*DEL:" [" do-net"] THEN RETURN(TRUE)
    ELSE RETURN(FALSE);                                                 02069
```

```
    END.                                                                02070
% Online deliveY CITATION FORMAT /                                     04148
  (olhed) PROC (adrstid, mflag, stid, l1, l2, recorded, wrkstr);        02729
    %Format stid to online delivery citation format%                    02730
    %l1, l2 delimit location of document or message (link)%             02731
    %leaves citation in lit and the comment, if any, in lit2%           02732
    %mflag = TRUE for message; recorded = TRUE if catalogued/archived
    %                                                                   02749
    %Note: message text will appear as separate plex below citation.
    %                                                                   02733
    LOCAL TEXT POINTER z1;                                              02734
    REF l1, l2, wrkstr;                                                 02735
    ON SIGNAL ELSE                                                      02736
      IF mflag THEN RETURN(FALSE);                                      02737
    CCPOS SF(stid);                                                     02738
    FIND ↑z1;                                                           02739
    *lit* ← *[hjournal($z1, &wrkstr)]*,                                 02740
      EOL, *[jtitle($z1, &wrkstr)]*,                                    02741
      EOL, *[hcopyl($z1, &wrkstr)]*,                                    02742
      *[olloc(mflag, recorded, &l1, &l2, &wrkstr)]*,                    02743
      EOL, *[jnote(adrstid, &wrkstr)]*;                                 02744
    % set up text pointer for getjcomment. %                            02945
      jwp1 ← z1; jwp1[1] ← z1[1];                                       02946
    IF [getjcomment(&wrkstr)].L <= 0 THEN                               02745
      *wrkstr* ← NULL                                                   02746
    ELSE                                                                04269
      *wrkstr* ← "Comments: ", *wrkstr*;                                04270
    RETURN(TRUE);                                                       02747
    END.                                                                02748
(hjournal)PROC(z1, wrkstr);                                              0996
    LOCAL TEXT POINTER z2 , z3, z4, z5;                                  0997
    REF z1, wrkstr;                                                      0998
    CCPOS z1;                                                            0999
    IF FIND [".HJOURNAL="] ['"]  ↑z2 ['"] ↑z3 ←z3 THEN                  01000
      BEGIN                                                              01001
      IF FIND BETWEEN z2 z3([".SPLIT;"] ↑z4 < ['.] ↑z5 >) THEN          01002
        *wrkstr* ← z4 z3, " [", z2 z5, ']                                01003
      ELSE *wrkstr* ← z2 z3                                              01004
      END                                                                01005
    ELSE *wrkstr* ← NULL;                                                01006
    RE URN(&wrkstr)                                                      01007
    END.                                                                 01008
(jtitle)PROC(z1, wrkstr);                                                01009
    LOCAL TEXT POINTER z2 , z3;                                          01010
    REF z1, wrkstr;                                                      01011
    CCPOS z1;                                                            01012
    IF FIND ["Title: "] ['"] ↑z2 ['"] ↑z3 ←z3 THEN                       01013
      *wrkstr* ← z2 z3                                                   01014
    ELSE                                                                 01015
      *wrkstr* ← NULL;                                                   01016
    RETURN(&wrkstr)                                                      01017
    END.                                                                 01018
```

```
(hcopy1)PROC(z1, wrkstr);                                              01019
   REF z1, wrkstr;                                                     01020
   CCPOS z1;                                                           01021
   IF FIND ["Hard Copy--Location: "] THEN                              01022
      *wrkstr* ← "Hard Copy Only--"                                    01023
   ELSE                                                                01024
      *wrkstr* ← NULL;                                                 01025
   RETURN(&wrkstr)                                                     01026
   END.                                                                01027

(olloc)PROC(mflag, recorded, z1, z2, wrkstr);                          02721
   REF z1, z2, wrkstr; % the link %                                    02722
   IF mflag AND NOT recorded THEN                                      02723
      *wrkstr* ← "Message: " % no link %                               02724
   ELSE                                                                02725
      *wrkstr* ← "Location: ",  z1 z2;                                02726
   RETURN(&wrkstr)                                                     02727
   END.                                                                02728

(gtollink)PROC(z1, z2);                                                02107
   REF z1, z2;                                                         02108
   RETURN(FIND SF(*lit*) [EOL] [EOL] ["Location: "] ['('] ↑z1 ←z1
   [')'] ↑z2);                                                         02110
   END.                                                                02109

(jnote)PROC(stid, wrkstr);                                             01037
   % avoid putting out special action indicator as a note %           02567
   LOCAL TEXT POINTER z1, z2;                                          01038
   REF wrkstr;                                                         02944
   IF FIND SF(stid) ["Note: "] ↑z1 [";;"] ↑z2 ←z2 ←z2 THEN             04271
      BEGIN                                                            04272
      *wrkstr* ← z1 z2;                                                04273
      IF FIND SF(*wrkstr*) ↑z1 (" [ ACTION ] "/" [ INFO-ONLY ] ")
      ↑z2 THEN                                                         04278
         BEGIN                                                         04279
         IF FIND z2 ') THEN                                            04280
            *wrkstr* ← NULL                                            04281
         ELSE                                                          04282
            ST z1 z2 ← NULL;                                           04283
         END;                                                          04284
      IF wrkstr.L THEN                                                 04274
         *wrkstr* ← "*****Note: ", *wrkstr*, "*****", EOL;             04275
      END                                                              04277
   ELSE *wrkstr* ← NULL;                                               04265
   RETURN(&wrkstr);                                                    01042
   END.                                                                01043

% Control support routines: check for ↑S, full disc, load averages too
high %                                                                 03389
   (discfull)PROCEDURE(pgs);                                           03312
      %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. %                            03313
      lavchk();                                                        03314
      %check disc pages%                                               03315
```

```
      r1←600000777777B;                                                  03316
      !JSYS gdskc; %214B%                                                03317
      IF pgs>r2 THEN RETURN(TRUE);                                       03318
   RETURN(FALSE);                                                        03319
   END.
                                                                         03320
(lavchk)PROC; %dismiss until load average < oljmlav; also, check if
control S has been typed.%                                               03321
   LOCAL sdstoptime;                                                     03322
   %convenient breakpt on ↑S; this is called from a critical
   location%                                                             03323
      !SKIPE inpstp; stpchk();                                           03324
   %dismiss if load average is too high, terminate process if system
   going down soon%                                                      03325
      LOOP                                                               03326
         BEGIN                                                           04115
         IF SKIP !getab(lavtabad) AND r1 > oljmlav THEN BEGIN            03327
            %dismiss%                                                    03328
            r1 ← getsbn($"OJHLAV");                                      03329
            !JSYS setnm;                                                 03330
            r1 ← 100000; !JSYS disms; %100 seconds%                      03331
            r1 ← oljsbn; !JSYS setnm; %restore subsystem name%           03332
            REPEAT LOOP;                                                 03333
            END;                                                         03334
         IF !gtad() > nsdcktime THEN                                     03335
            BEGIN %haven't checked for 5 minutes%                        03336
            gtadad5();                                                   03337
            nsdcktime ← r1; %gtad time 5 minutes from now%               03338
            gtadad5(); sdstoptime ← r1;                                  03339
            r1 ← lavtabad; r1.LH ← 27B;                                  03340
            IF SKIP !getab() AND r1 AND sdstoptime > r1 THEN SIGNAL
            (sigdsf, $"Terminated:  TENEX going down within 10
            minutes");                                                   03341
            END;                                                         03342
         EXIT LOOP;                                                      03343
         END;                                                            03344
   RETURN;                                                               03345
   END.
                                                                         03346
(stpchk)PROC; %got ↑S; terminate%                                        03352
   % Check if user really wants to terminate.  If not, continue.
   The user interaction may not work with CML!. %                        04116
   LOCAL pofile;                                                         03353
   inpstp ← 0;                                                           03354
   !gpjfn(4B5);                                                          03355
   IF (pofile ← r2.RH) # 777777B THEN                                    03356
      BEGIN                                                              03357
      r2.RH ← 777777B;                                                   03358
      !spjfn();                                                          03359
      END;                                                               03360
   (stpcloop):                                                           03361
   ON SIGNAL                                                             03362
      =sigdsf: NULL;                                                     03363
      ELSE GOTO stpcloop;                                                03364
   typeas($"↑S interrupt:  Terminate or Continue? ");                    03365
   CASE inpcuc() OF                                                      03366
```

```
        ='T:                                                          03367
            BEGIN                                                     04119
            typeas($"erminate");                                      04118
            IF inpcuc() # CA THEN                                     04120
               BEGIN                                                  04121
               typeas($" ? ");                                        04122
               REPEAT CASE;                                           04123
               END;                                                   04124
            SIGNAL (sigdsf, $"TS Terminated");                        03369
            END;                                                      03370
        ='C:                                                          03371
            BEGIN                                                     04125
            typeas($"ontinue");                                       04126
            IF inpcuc() # CA THEN                                     04127
               BEGIN                                                  04128
               typeas($" ? ");                                        04129
               crlf();                                                04134
               REPEAT CASE;                                           04130
               END;                                                   04131
            END;                                                      04133
        ENDCASE                                                       03372
            BEGIN                                                     04135
            typeas($"?");                                             03373
            crlf();                                                   04136
            REPEAT CASE;                                              03374
            END;                                                      03375
      ON SIGNAL ELSE NULL;                                            03376
      IF pofile # 777777B THEN                                        03377
         BEGIN                                                        03378
         !gpjfn(hB5);                                                 03379
         r2.RH ← pofile;                                              03380
         !spjfn();                                                    03381
         END;                                                         03382
      RETURN;                                                         03383
      END.
                                                                      03384
   (gtadad5)PROC; %add 5 minutes to gtad time in r1%                  03347
      !ADDI r1,527654B;  %(186-(3600*24))+300%                        03348
      !TRNE r1,400000B;                                               03349
      !SUBI r1,527200B;  %didn't go through 2400 GMT%                 03350
      RETURN; END.
                                                                      03351
% Utility routines: driver file timing table setting and reading
procedures; time checking %                                           04137
   (jsettitab)PROCEDURE;                                              03457
      % 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. %                 04104
      LOCAL jttstid, stid, i, mlav[2];                                03458
      LOCAL STRING lavstr[30];                                        03459
      LOCAL TEXTPOINTER z1, z2;                                       03460
      i ← 0;                                                          03462
      stid ← origin;                                                  03463
      ON SIGNAL ELSE                                                  03461
         snkerr($"Driver File open fail");                            04181
```

```
   stid.stfile ← open(0, jflname($"jdriver"));                          03464
   ON SIGNAL ELSE close(stid.stfile);                                   04180
   IF (jttstid ← namelook(stid, $"timetable")) = endfil THEN            03466
      snkerr ($"No timetable in driver file");                          04105
   stid ← getsub(jttstid);                                              03467
   WHILE stid # jttstid DO                                              03468
      jnltime [i:=i+1] ← setjtw (stid:=getsuc(stid));                   04106
   jnltime[i] ← 0; %terminator%                                         03469
   IF NOT liblod % load average set by hand % THEN                      03470
      BEGIN                                                             03471
      IF (stid ← namelook(jttstid, $"mlav")) = endfil THEN snkerr
      ($"No mlav statement in driver file");                            03472
      FIND SF(stid) [')] [(':/)] ↑z1 [(';<CH / ENDCHR)] ↑z2;            03473
      *lavstr* ← z1 z2;                                                 03474
      nfloat($lavstr, $mlav, $mlav+1);                                  03475
      oljmlav ← mlav;                                                   03476
      END;                                                              03477
   close(jttstid.stfile);                                               03478
   RETURN;                                                              03479
   END.                                                                 03480

(setjtw) PROCEDURE(stid); %process one statement to make one table
word for jnltime%                                                       03250
   LOCAL char, fchar, jnte, code;                                       03251
   LOCAL STRING tempsr[200];                                            03252
   jnte ← 0;                                                            03253
   *tempsr* ← SF(stid) SE(stid);                                        03254
   astruc($tempsr);                                                     03255
   FIND SF(*tempsr*);                                                   03256
   LOOP CASE (fchar ← READC) OF                                         03257
      ='P, ='N:                                                         03258
         BEGIN %Set the Period in hours into the INTERVAL (OJTIII)
         field; the Number of intervals into the NUMBER (OJTINN)
         field%                                                         03259
         r1 ← 0;                                                        03260
         FIND [':];                                                     03261
         WHILE (char ← READC) # '; DO                                   03262
            IF char IN ['0, '9] THEN r1 ← r1*10 + (char -'0);           03263
         CASE fchar OF                                                  03264
            ='P: jnte.ojtiii ← r1;                                      03265
            ='N: jnte.ojtinn ← r1;                                      03266
            ENDCASE;                                                    03267
         END;                                                           03268
      ='T:                                                              03269
         BEGIN % Set the time into the TIME (OJTITT) field%             03270
         r1 ← 0;                                                        03271
         FIND [':];                                                     03272
         WHILE (char ← READC) # '; DO                                   03273
            BEGIN                                                       03274
            IF char IN ['0, '9] THEN r1 ← r1*10 + (char -'0);           03275
            END;                                                        03276
         !IDIVI r1,100; %hours now in r1, minutes in r2%                03277
         !LSH r1,6;                                                     03278
         jnte.ojtitt ← r1 + ((r2 *64) + 30)/60; %time in 1/64ths of
         hour%                                                          03279
         END;                                                           03280
```

```
      ='D: % Set the tasks to be done at the appropriate wake up
      into the JFLG (OJTIJJJ) field%                                    03281
         BEGIN                                                          03282
         FIND (':);                                                     03283
         code ← 0;                                                      03284
         LOOP CASE READC OF                                             03285
            ='O: code ← code .V 2B; % On-line distribution. %           03286
            ='S: code ← code .V 4B; % Slinker. %                        03287
            ='C: code ← code .V 10B; % Run detached, go to sleep,
            awake and continue processing. %                            03288
            ='H: code ← code .V 20B; % Run Hard copy distribution. %
                                                                        03289
            =';: EXIT LOOP;                                             03290
            ENDCASE;                                                    03291
         jnte.ojtijjj ← code;                                           03292
         END;                                                           03293
      =ENDCHR:  RETURN(jnte);                                           03294
      ENDCASE NULL;                                                     03295
   END.
                                                                        03296
(getojtime) PROCEDURE (ctime);                                          03199
   %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%                                  03200
   %Format of table jnltime entry: jjjiinnhhffB                         03201
      jjj: option flags: see joutil for meaning;  becomes libflg
                                                                        04107
      ii:  period (interval) length in hours                            04108
      nn: number of deliveries (at ii-hour intervals)                   03202
      hh: hours field in time                                           03203
      ff: 64ths of hour field in time                                   03204
   %                                                                    03205
   LOCAL jcnt, jnte, jntes, val, btime, dcnt, inc;                      03206
   jcnt ← 0;                                                            03207
   val ← 777777B;                                                       03208
   WHILE (jnte ← jnltime (jcnt := jcnt + 1)) # 0 DO                     03209
      BEGIN                                                             03210
      btime ← ((jnte.ojtitt) * 60) / 64;                                03211
      dcnt ← (jnte.ojtinn) .A 77B;                                      03212
      IF NOT (inc ← jnte.ojtiii) THEN inc ← 1;                          03213
      inc ← inc * 60;                                                   03214
      DO                                                                03215
         BEGIN                                                          04102
         IF btime > ctime AND btime < val THEN                          03216
            BEGIN                                                       03217
            val ← btime;                                                03218
            jntes ← jnte;                                               03219
            END;                                                        03220
         END                                                            03221
      UNTIL (dcnt ← dcnt - 1) <= 0 OR (btime := btime + inc) >
      ctime;                                                            04103
      END;                                                              03222
   IF val = 777777B THEN val ← jntes ← 0;                               03223
   RETURN(val % the time %, jntes.ojtijjj % The Journal utility flag
   %);                                                                  03224
   END.
```

```
                                                                    03225
                                                                    02825
(timcheck)PROC;                                                     02826
   LOCAL cnt;                                                       02827
   cnt ← 0;                                                         02828
   LOOP                                                             02829
      BEGIN                                                         02830
      !JSYS gtad;                                                   02831
      IF r1 # -1 THEN RETURN;                                       02832
      IF (cnt:=cnt-1) <=0 THEN                                      02833
         BEGIN
         specttyout(oprtty, $"System does not have Date and Time
                                                                    02834
         set.");
         cnt ← 20; %complain every 20 minutes%                      02835
         END;                                                       02836
      r1 ← 60000; %one minute%                                      02837
      !JSYS dismS;                                                  02838
      END;                                                          02839
   END.                                                             02840
                                                                    02841
                                                                    04138
% Error handling.  Termination. %                                  03297
  (snkerr)PROC(astr);
     %Accepts an error string in astr, blaps it to tty0, tty30, and
     primary output, an halts%                                      03298
     REF astr;                                                      03299
     %First set up subsystem name to snkerr%                        03300
        r1 ← getsbn($"SNKERR");                                     03301
        !JSYS setnm;                                                03302
     %Now blap error to tty 0 and 30%                               03303
        specttyout(0, &astr);                                       03304
        specttyout(30B, &astr);                                     03305
     %now type it to primary output%                                03306
        crlf();                                                     03307
        typeas(&astr);                                              03308
     %Now die%                                                      03309
        !JSYS haltf;                                                03310
     END.                                                           03311
                                                                    04141
% Network mail utility procedures. %
  (pokemailer)PROC(dirstr); %set bit in system file to tell mailer to
  get busy%                                                         02112
     LOCAL indx, blkadr, dirno, jfn;                                02136
     LOCAL STRING tempsr[70];                                       02137
     REF dirstr;                                                    02155
     jfn ← 0;                                                       02141
     ON SIGNAL ELSE                                                 02138
        BEGIN                                                       02139
        IF jfn THEN sysclose(jfn, $tempsr);                         02140
        RETURN (FALSE);                                             02142
        END;                                                        02143
     %get directory number%                                         02149
        IF &dirstr AND dirstr.L > 0 THEN                            02180
           BEGIN                                                    02181
           *tempsr* ← *dirstr*;  %transdir adds a "0" to the string,
           so--%                                                    02156
           dirno ← transdir($tempsr);                               02150
```

```
        END                                                          02182
      ELSE                                                           02183
        BEGIN %use connected directory%                              02184
        !JSYS gjinf;                                                 02186
        dirno ← r2;                                                  02187
        END;                                                         02185
   %open mailer's flag file%                                         02126
      jfn ← sgtjfn( getgtjflg(read,0,oldvrsn),                       02115
      $"<SYSTEM>MAILER.FLAGS;1", $tempsr );                          02116
      sysopen (jfn, rwthawed, random, $tempsr);                      02124
   %get a frozen core page%                                          02121
      frzblk ((indx ← lodrfb(0,-1)), 1);                             02122
      blkadr ← crpgad[indx];                                         02127
   % map in file page%                                              02117
      !HRLZ r1,jfn;                                                  02118
      r2 ← 4B11 .V (blkadr/1000B);                                   02119
      r3 ← 14B10;                                                    02120
      !JSYS pmap;                                                    02128
   %set the bit%                                                     02129
      r3 ← dirno;                                                    02130
      !IDIVI r3,44B;                                                 02131
      r1 ← 4B11;                                                     02132
      r4 ← -r4;                                                      02135
      r3 ← r3 + blkadr;                                              02133
      !ROT r1,0(r4);                                                 02134
      !IORM r1,0(r3);                                                02144
   %map out the page%                                                02145
      r1 ← -1; %r2 is still set%                                     02146
      !JSYS pmap;                                                    02147
   %close mailer's file%                                             02148
      sysclose(jfn, $tempsr);                                        02125
   %thaw core page%                                                  02123
      frzblk(indx, -1);                                              02114
   RETURN(TRUE);                                                     02113
   END.                                                              02022

(openmail)PROC(userstr, hoststr, dirstr);                            02172
   LOCAL STRING fnamstr[150];                                        02178
   REF userstr, hoststr, dirstr;                                     02173
   %make up name%
      IF &dirstr AND dirstr.L > 0 THEN *fnamstr* ← '<, *dirstr*, '>  02174
      ELSE *fnamstr* ← NULL;                                         02175
      *fnamstr* ← *fnamstr*, "[--UNSENT-MAIL--].", *userstr*;        02176
      IF hoststr AND hoststr.L > 0 THEN *fnamstr* ← *fnamstr*, 26B,  02050
      '@, *hoststr*;                                                 02051
   %get stack and switch to it%                                      02088
      &ojsqsw←getsgw();                                              02052
      ojsqsw.swstid ← $fnamstr;                                      02053
      s ← ojsqsw.swsrsav := s;
      m ← ojsqsw.swmrsav := m;
      %calling parameters and locals can not be used while the stack
      is switched%                                                   02177
   ON SIGNAL ELSE                                                    02089
      BEGIN                                                          02090
      tda.dausqcod← 0;                                               02092
      tda.davspec.vsusqf ← FALSE;                                    02093
```

```
      ojsqsw.swstid ← FALSE;                                          02095
      sport(&ojsqsw); %return to the close-out call of sport in
      closemail%                                                      02094
      END;                                                            02091
   tda.dausqcod← ojsqlink;                                            02054
   tda.davspec.vsusqf ← TRUE;                                         02055
   %the following call to opseqf causes the first port call through
   ojsqlink  which looks like a return to the caller of openmail%
                                                                      02061
      opseqf( ojsqsw.swstid, &tda, FALSE, FALSE, opmlfl);             02056
   %the return to this point is affected by the final port call in
   closemail%                                                         02062
   tda.dausqcod← 0;                                                   02057
   tda.davspec.vsusqf ← FALSE;                                        02058
   ojsqsw.swstid ← FALSE;                                             02096
   sport(&ojsqsw); %return to the close-out call of sport in
   closemail%                                                         02059
   END.
                                                                      02023
(mailstring)PROC(str, lev); %send a string to output sequential at
given level%                                                          02024
   ojsqsw.swstid ← asrref(str);                                       02044
   ojsqsw.swclvl ← lev;                                               02046
   sport (&ojsqsw); %port to ojsqlink%                                02045
   RETURN;                                                            02049
   END.
                                                                      02025
(ojsqlink)PROC(sw, calltype); %link between journal process and
output sequential%                                                    02032
   REF sw;                                                            02034
   CASE calltype OF                                                   02035
      =sqgnxt: NULL;                                                  02036
      ENDCASE RETURN;                                                 02037
   LOOP                                                               02038
      BEGIN                                                           04142
      ojsqsw.swstid ← TRUE; %return true%                             02097
      sport(&ojsqsw); %port to journal process%                       02039
      sw.swstid ← ojsqsw.swstid; %journal process' astring on its
      way to output sequential%                                       02040
      sw.swclvl ← ojsqsw.swclvl;                                      02047
      sport(&sw); %port to output sequential%                         02041
      END;                                                            02042
   END.
                                                                      02033
(closemail)PROC;                                                      02026
   ojsqsw.swstid ← endfil;                                            02028
   sport(&ojsqsw); %final port call into output sequential%           02029
   %is now all closed out, release seq work area%                     02043
   relsgw(&ojsqsw);                                                   02030
   RETURN;                                                            02031
   END.
                                                                      02027
(ftprcop)PROC(hostnam, hldir, hlpassw, hext, dirnam);                 04286
   LOCAL pos, jfn, errflg, ftstate, oftstate, checkq, filspec, ojfn;
                                                                      04287
   LOCAL STRING ifilnam[80], ofilnam[80], xfilnam[80], qfilnam[80],
```

```
    sqfilnam[80], nfilnam[80], lext[10], tempsr[80], jftperm[300];          04288
                                                                            04289
    LOCAL TEXT POINTER p1, p2, p3, p4;                                      04290
    REF hostnam, hldir, nlpassw, hext, dirnam;                              04291
    checkq ← 0;                                                             04292
    ojfn←0;                                                                 04293
    jfn ← 0;                                                                04294
    errflg ← 0;                                                             04295
    ftstate←oftstate←0;                                                     04296
    FIND SF(*hext*) ↑p1 ([';] / [ENDCHR]) ↑p2;                              04297
    *lext* ← p1 p2;                                                         04298
    (ftprstart):                                                            04299
    ON SIGNAL ELSE                                                          04300
       BEGIN                                                                04301
       IF jfn THEN sysclose (jfn:=0, 0);                                    04302
       ON SIGNAL ELSE;                                                      04303
       ftpcls();                                                            04304
       ftpend();                                                            04305
       *jftperm* ← "Journal FTP err: ", *[sysmsg]*;                          04306
       IF NOT ftstate THEN RETURN;                                          04307
       IF ftstate = 2 AND NOT SKIP !delf(ojfn) THEN NULL;                   04308
       IF ftstate = 3 THEN                                                  04309
          BEGIN                                                             04310
          %remote rename error, but file got accross ok%                    04311
          %must also test for local NLQE files%                             04312
          jrnamf(ojfn, $ofilnam, $"NLQE;");                                 04313
          *jftperm* ← *jftperm*, EOL, "rename error on remote file ",
          *ifilnam*, EOL, "local file is ", *ofilnam*, EOL, "
          HELP!!";
          %type message to operator unless it's going to be done
          below%                                                            04314
             IF NOT errflg THEN specttyout(oprtty, $jftperm);               04315
          END;                                                              04316
       reljfn(ojfn); %better check this is ok for zero jfn%                 04317
       IF (errflg:=errflg+1) THEN %error abort, could test ftstate
       too%                                                                 04318
          BEGIN                                                             04319
          IF ftstate=2 AND oftstate=2 THEN                                  04320
             BEGIN                                                          04321
             *jftperm* ← *jftperm*, EOL, "happened twice on file: ",
             *ifilnam*;                                                     04322
             IF NOT checkq THEN *sqfilnam* ← *ifilnam*; %was xfering
             file%                                                          04323
             END;                                                           04324
          %type message to operator%                                        04325
             specttyout(oprtty, $jftperm);                                  04326
          RETURN;                                                           04327
          END;                                                              04328
       oftstate←ftstate;                                                    04329
       GOTO ftprstart;                                                      04330
       END;                                                                 04331
    ftpbgn();                                                               04332
    ftpopn (&hostnam);                                                      04333
    ftplog(&hldir, &hlpassw, $"103");                                       04334
    LOOP BEGIN                                                              04335
       ftstate←1;                                                           04336
```

```
        filspec ← IF checkq THEN $"*.nlq;*" ELSE $"*.nlr;*";                04337
        ftpdir($"djunk.txt;1", filspec);                                    04338
        IF NOT jfn ← lgetjfn (0, $"djunk.txt;1", $txtext, gtjoisf,
        $lit) THEN                                                          04339
           callsig (ofilerr  $lit);                                         04340
        IF NOT sysopen (jfn, read, chrtyp, $lit) THEN                       04341
           BEGIN                                                            04342
           reljfn (jfn);                                                    04343
           callsig (ofilerr, $lit);                                         04344
           END;                                                             04345
        WHILE isread (jfn, $ifilnam, LF) DO                                 04346
           BEGIN                                                            04347
           ftstate←1;                                                       04348
           IF NOT (FIND SF(*ifilnam*) $LF ↑p1 ( ('< ['>] ) / ) ↑p4
           ['.] ↑p2 [';] [';] ↑p3 ←p3) THEN EXIT;                           04349
           *ifilnam* ← p1 p3;                                               04350
           *qfilnam* ← p1 p2, "NLQ";                                        04351
           IF NOT checkq AND sqfilnam.L#0 AND *sqfilnam* = *ifilnam*
           THEN                                                             04352
              BEGIN                                                         04353
              ftpren ($ifilnam, $qfilnam);                                  04354
              REPEAT LOOP;                                                  04355
              END;                                                          04356
           *xfilnam* ← p1 p2, *lext*;                                       04357
           *ofilnam* ← *dirnam*, p4 p2, "NLQ";                              04358
           IF NOT (ojfn ← sgtjfn(600101B6, $ofilnam, $tempsr)) THEN
           callsig(ofilerr, $tempsr);                                       04359
           jfnstr(ojfn, $ofilnam);                                          04360
           ftstate←2;                                                       04361
           ftprtr($ofilnam, $ifilnam, $"COMPRESSED");                       04362
           ftstate←3;                                                       04363
           ftpren ($ifilnam, $xfilnam);                                     04364
           ftstate←4;                                                       04365
           reljfn(jrnamf(ojfn, $ofilnam, $"NLR;"));                         04366
           errflg←ftstate←0;                                                04367
           END;                                                             04368
        IF jfn THEN sysclose(jfn:=0, 0);                                    04369
        IF (checkq:=checkq+1) THEN EXIT LOOP;                               04370
        END;                                                                04371
     ftpcls();                                                              04372
     ftpend();                                                              04373
     RETURN;                                                                04374
     END.                                                                   04375
% Miscellaneous %                                                          04139
   (getnhcflag)PROC(submachine);                                            02844
      LOCAL flgs;                                                           02845
      flgs ← 0;                                                             02846
      IF submachine THEN flgs.delnet ← TRUE;                                02847
      IF lhostn=utilhost THEN flgs.delhc ← TRUE;                            02848
      RETURN(flgs);                                                         02849
      END.                                                                  02850

   (setmastac)PROC; %set mastacprintflag; called by initialization
   routine.%                                                                02851
      mastacprintflag ← IF lhostn = utilhost THEN TRUE ELSE FALSE;          02852
      RETURN; END.
```

```
                                                                    02853
(jrnamf)PROC(jfn,filenm,newext); % rename journal file with new
                                                                    05
extension. %
   %rename file with extension newext.  renames file with jfn "jfn"
   unless "jfn"=0, then uses name "filenm".  returns new string in
                                                                    06
   "filenm".%                                                       07
   LOCAL newjfn;                                                    08
   LOCAL STRING tempsr [50];                                        09
   LOCAL TEXT POINTER z1, z2;                                       010
   REF filenm, newext;
   IF jfn THEN jfnstr(jfn,&filenm) ELSE jfn ← sgtjfn(100101B6,
                                                                    011
   &filenm,$tempsr);                                                012
   FIND SF(*filenm*) ['.] tz1 [ENDCHR] tz2;                         013
   ST z1 z2 ← *newext*;
   IF NOT newjfn←sgtjfn(600101B6, &filenm,$tempsr) THEN
                                                                    016
   snkerr($tempsr);                                                 015
   r2←newjfn;                                                       016
   r1←jfn;                                                          017
   IF NOT SKIP !JSYS rnamf %35B% THEN BEGIN                         018
      *tempsr* ← "RENAME err-file ", *filenm*;                      019
      snkerr($tempsr);                                              020
      END;
   RETURN (newjfn); END.                                            021

                                                                    04146
% Ident list expansion %
 (expdist)PROC(distid, p1); %expand idents pointed to by p1 into
                                                                    01187
 statement(s) down from distid%                                     01188
   LOCAL distexf, idfilno, going;                                   01190
   LOCAL STRING dstring[700];                                       01205
   REF p1;                                                          01194
   idfilno ← open(0, jflname($"identfile"));                        01195
   *dstring* ← NULL;                                                01196
   intids();                                                        01197
   DO BEGIN
      WHILE (going ← getids (&p1, $dstring, 1, idfilno)) AND
      dstring.L < 650 DO *dstring* ← *dstring*, ", ";               01198
      IF distexf THEN *dstring* ← *dstring*, ".PES;.HLT;";          01199
      distexf ← FALSE;                                              01200
      cis(distid, $dstring, $downstr);                              01201
      *dstring* ← NULL;                                             01202
      END UNTIL NOT going;                                          01203
   close (idfilno);                                                 01204
   RETURN;                                                          01191
   END.                                                             01192

                                                                    01193
 (iexpdist)PROC(delstr,idfnum);                                     01469
   LOCAL wrkstr;                                                    02947
   LOCAL TEXT POINTER ptr, z1, z2;                                  01470
   REF delstr, wrkstr;                                              01471
   &wrkstr ← getstring(1000, $dspblk);                              02948
   *delstr* ← *delstr*, ';;                                         01454
   makeptr(asrref(&delstr),$ptr);                                   01455
   *wrkstr* ← " ";                                                  01456
   intids(0);                                                       01457
   LOOP                                                             01458
```

```
        BEGIN                                                          01459
        *lit* ← NULL;                                                  01460
        IF NOT getids($ptr, $lit, 0, idfnum) THEN EXIT;                01461
        getiid($lit, 0, $z1, $z2);                                     01462
        *wrkstr* ← *wrkstr*, z1 z2, " ";                               01463
        END;                                                           01464
     *delstr* ← *wrkstr*;                                              01474
     freestring(&wrkstr, $dspblk);                                     02949
     RETURN;                                                           01473
     END.                                                              01472
                                                                       01475
FINISH jnldel                                                          01125
% Error handling Routines-- currently unused%                          03987
   (jerror)PROC(errval);                                               03986
      LOCAL count;                                                     03989
      jclosfl();                                                       03990
      IF errval < 0 THEN                                               03991
         BEGIN                                                         03992
         r1 ← chbptr(empty) + $errwrk;                                 03993
         r2 ← -errval .V 4B11;                                         03994
         r3 ← 0;                                                       03995
         !JSYS erstr;                                                  03996
         GOTO baderr;                                                  03997
         GOTO baderr;                                                  03998
         count ← 0;                                                    03999
         (baderr): dismes(2,$"System Error");                          04000
         END                                                           04001
      ELSE dismes(2,errval);                                           04002
      nlsrst();                                                        04003
      END.                                                             04004
%Procedures for initialising Journal Entry Mode-- currently unused%    04005
   (jntrint) PROCEDURE;                                                04006
      %This procedure checks the jcurnal flags and sets the system name
      upon entrance to the journal.%                                   04007
      jdirn ← 0;                                                       04008
      IF jdebug THEN                                                   04009
         BEGIN                                                         04010
         dismes(1, $"Experimental Journal System--Do not use");        04011
         END                                                           04012
      ELSE                                                             04013
         IF jolock() THEN                                              04014
            BEGIN                                                      04015
            IF nlmode = fulldisplay THEN dismes(1, $"Deferred numbers
            only") ELSE typeas ($"(Deferred numbers only) ");          04016
            END;                                                       04017
      %set up subsystem name for statistics%                           04018
         IF NOT jnlsbn THEN jnlsbn ← <AUXCOD, getsbn>($jnlnam);        04019
         r1 ← jnlsbn;                                                  04020
         !JSYS setnm;                                                  04021
      RETURN;                                                          04022
      END.                                                             04023
   (jctlcint) PROCEDURE;                                               04024
```

```
    % This procedure performs the initialization required to service
    control C in the program.%                                          04025
    r1 ← 4B5;                                                           04026
    !JSYS rpcap;                                                        04027
    r3 ← r2;                                                            04028
    !JSYS epcap; %Permits process to assign ↑C for pseudo interrupt%    04029
                                                                        04030
    r1.LH ← 3;                                                          04031
    r1.RH ← 2;                                                          04032
    !JSYS ati;                                                          04033
    r1 ← 4B5;                                                           04034
    r2 ← 1B11;                                                          04035
    !JSYS aic; %Activate ↑C interrupt channel = 2%                      04036
    conjdir(TRUE);
    jt0 ← jt1 ← jt2 ← jt3 ← jt4 ← jt5 ← jt6 ← jt7 ← jt8 ← jt9 ← jt10
    ← jt1 ← jt12 ← jt13 ← jt14 ← jt15 ← 0;                              04037
    RETURN;                                                             04038
    END.                                                                04039

%Routines for wrapping up Journal and returning to NLS-- currently
unused%                                                                 04040
  (jreset)PROC;                                                         04041
    jclosfl();                                                          04042
    %toff 13%                                                           04043
    IF jtimfg THEN jtime();                                             04044
    nlsrst();                                                           04045
    END.                                                                04046

%Timing procedures-- currently unused%                                  04047
  (jtime)PROC;                                                          04048
    %Type out jthe jtimes of jthe run, and wait for a command accept
    before proceeding%                                                  04049
    jtimout($"Open File", jt0);                                         04050
    jtimout($"Open Lock", jt1);                                         04051
    jtimout($"Open Work", jt2);                                         04052
    jtimout($"Set J Header", jt3);                                      04053
    jtimout($"Find Active", jt4);                                       04054
    jtimout($"enter J dcc", jt5);                                       04055
    jtimout($"Update & delfn", jt6);                                    04056
    jtimout($"Update JCAT", jt7);                                       04057
    jtimout($"Jcat Entry", jt8);                                        04058
    jtimout($"Dist Doc", jt9);                                          04059
    jtimout($"check Ident", jt11);                                      04060
    jtimout($"Set J Work", jt12);                                       04061
    jtimout($"Get Number", jt10);                                       04062
    jtimout($"Used Numbers", jt14);                                     04063
    jtimout($"Check Number", jt15);                                     04064
    jtimout($"Summed Total ", (jt0+jt1+jt2+jt3+jt4+jt5+jt6+jt7)
    +(jt8+jt9+jt10+jt11+jt12+jt14+jt15));                               04065
    jtimout($"Elapsed Total", jt13);                                    04066
    crlf();                                                             04067
    typeas($"(Type CA)");                                               04068
    WHILE input() # CA DO NULL;                                         04069
    RETURN END.                                                         04070

  (jtimout)PROC(astr, value);                                           04071
```

```
    REF astr;                                                          04072
    LOCAL STRING st[20];                                               04073
    crlf();                                                            04074
    typeas(&astr);                                                     04075
    typeas($" = ");                                                    04076
    *st* ← STRING(value);                                              04077
    typeas($st);                                                       04078
    RETURN END.                                                        04079

%Katalog Klean-up-- currently unused%                                  03640
  (getcfn) %fetch filename from catalog entry%
  PROCEDURE (entrystid, fldstr, ptr1, ptr2);                           03618
    %----------%                                                       03619
    LOCAL linkstid, retval;                                            03620
    LOCAL STRING dir [40], stmt [40], vspecs [40];                     03621
    LOCAL TEXT POINTER p1, p2;                                         03622
    REF fldstr;                                                        03623
    %----------%                                                       03624
    retval ← FALSE;                                                    03625
    IF NOT &fldstr THEN err ();                                        03626
    linkstid ← getsub (entrystid);                                     03627
    IF NOT (FIND SF(linkstid) ["Link"] ['('] ↑p1 ←p1 [')'] ↑p2) THEN   03628
      getcend (entrystid, $p1, p2)                                     03629
    ELSE                                                               03630
      BEGIN                                                            03631
      lnkspec (1, $p1, $dir, &fldstr, $stmt, $vspecs);                 03632
      *fldstr* ← *fldstr*, ".NLS";                                     03633
      retval ← TRUE;                                                   03634
      END;                                                             03635
    stptset (0, ptr1, ptr2, $p1, $p2);                                 03636
    RETURN (retval);                                                   03637
    END.                                                               03638

                                                                       03639
                                                                       03641
  (updtjo)PROC;
    %This procedure updates the pemanent copies of the Journal files
    JCAT an CNUMBERS from the temporary copies TCNUMBERS, TJCAT%       03642
    LOCAL jcstid, tjcstid, cstid, tcstid;                              03643
    sabtfg ← jcstid ← tjcstid ← cstid ← tcstid ← 0;                    03644
    jsavaccess ← access;                                               03645
    conjdir(TRUE);                                                     03646
    rl←getsbn($"JUPDOH");                                              03647
    !JSYS setnm;                                                       03648
    %**** STATE ← jcopst, spec; **** %                                 03649
    IF sabtfg THEN supervisor()                                        03650
    ELSE BUMP sabtfg;                                                  03651
    ON SIGNAL ELSE                                                     03652
      BEGIN                                                            03653
      undo(jcstid := 0);                                               03654
      undo(tjcstid := 0);                                              03655
      undo(cstid := 0);                                                03656
      undo(tcstid := 0);                                               03657
      access ← jsavaccess;                                             03658
      conjdir(FALSE);                                                  03659
      rl←nlssbn;                                                       03660
```

```
    !JSYS setnm;                                                    03661
    END;                                                            03662
  enablaccess(0, jrnlaccess);                                       03663
  %First do the JCAT file%                                          03664
    jcstid ← openlock(0, jflname($"JCAT"));                         03665
    tjcstid ← openlock(0, jflname($"TJCAT"));                       03666
    updtpx(jcstid, tjcstid, $"Jcatalog", FALSE, FALSE);             03667
    updtpx(jcstid, tjcstid, $"catmods", TRUE, FALSE);               03668
    closeu((jcstid:=0).stfile);                                     03669
    closeu((tjcstid:=0).stfile);                                    03670
  %Now do cnumbers%                                                 03671
    cstid ← openlock(0, jflname($"CNUMBERS"));                      03672
    tcstid ← openlock(0, jflname($"TCNUMBERS"));                    03673
    updtpx(cstid, tcstid, $"FREE", FALSE, FALSE);                   03674
    updtpx(cstid, tcstid, $"USED", FALSE, FALSE);                   03675
    updtpx(cstid, tcstid, $"INUSE", FALSE, FALSE);                  03676
    updtpx(cstid, tcstid, $"PREASSIGNED", FALSE, TRUE);             03677
    closeu((cstid:=0).stfile);                                      03678
    closeu((tcstid := 0).stfile);                                   03679
  access ← jsavaccess; %restore access%                             03680
  conjdir(FALSE);                                                   03681
  r1←nlssbn;                                                        03682
  !JSYS setnm;                                                      03683
  supervisor();                                                     03684
  END.                                                              03685

(updtpx)PROC(stid1, stid2, namest, tailf, replfg);                  03686
  LOCAL dir, tstid1, tstid2;                                        03687
  LOCAL TEXT POINTER z1, z2, z3, z4;                                03688
  REF namest;                                                       03689
  IF (tstid1 ← namelook(stid1, &namest)) = endfil THEN              03690
  badfil(stid1.stfile);
  IF (tstid2 ← namelook(stid2, &namest)) = endfil THEN              03691
  badfil(stid2.stfile);                                             03692
  IF (stid2 ← getsub(tstid2)) # tstid2 THEN                         03693
    BEGIN %there is something to copy%                              03694
    IF NOT (tailf OR replfg) THEN                                   03695
      BEGIN                                                         03696
      stid1 ← tstid1;                                               03697
      dir ← levdown                                                 03698
      END                                                           03699
    ELSE                                                            03700
      IF ((stid1 ← getsub(tstid1)) = tstid1) THEN %nothing
        under bracnn ... move to stid1 in levdown dir% dir ←        03701
        levdown                                                     03702
      ELSE                                                          03703
        BEGIN                                                       03704
        stid1 ← getail(stid1);                                      03705
        dir ← levsuc                                                03706
        END;                                                        03707
    IF replfg AND (stid1 # tstid1)  THEN
      crepgro(TRUE, gethed(stid1), getail(stid1), gethed(stid2),
      getail(stid2))                                                03708
    ELSE                                                            03709
      cmovgro(stid1, dir, gethed(stid2), getail(stid2), FALSE,
      0);                                                           03710
```

```
      END                                                                    03711
   ELSE                                                                      03712
      IF (stid1 ← getsub(tstid1)) = tstid1 THEN                              03713
         BEGIN                                                               03714
         FIND SF(stid1) ↑z1 SE(stid1) ↑z2;                                   03715
         FIND SF(stid2) ↑z3 SE(stid2) ↑z4;                                   03716
         creplex($z1, $z2, $z3, $z4);                                        03717
         END;                                                                03718
   RETURN END.                                                               03719

(undo)PROC(stid);                                                            03720
   %un-lock and close file%                                                  03721
   LOCAL fileno;                                                             03722
   IF (fileno ← stid.stfile) = 0 THEN RETURN;                                03723
   IF NOT writeout(fileno, $sar) OR NOT delpc(fileno, $sar)  THEN            03724

      err($sar);                                                             03725
   close(fileno);                                                            03726
   RETURN END.                                                               03727

(ncdex)PROCEDURE;                                                            01786
   LOCAL distcount, collcount, niccount, char, collections,
   distribution, nic, pcap;                                                  01787
   LOCAL TEXT POINTER z1;                                                    01788
   LOCAL STRING passw[5], jnum[5], tempsr[5];                                01789
   %Hard copy distribution execute%                                          01790
   setmastac();                                                              02843
   %Parse start-up command%                                                  01791
      %Password%                                                             01792
         *passw* ← NULL;                                                     01793
         echoff();                                                           01794
         crlf();                                                             01795
         prompt($"Password:");                                               01796
         UNTIL passw.L = 3 DO                                                01797
            IF (char ← inpcuc()) = CD THEN                                   01798
               GOTO STATE                                                    01799
            ELSE *passw* ← *passw*, char;                                    01800
         IF *passw* # "JPD" THEN                                             01801
            BEGIN                                                            01802
            tqmarks();                                                       01803
            GOTO STATE;                                                      01804
            END;                                                             01805
      % Get operators ident. %                                               01806
         crlf();                                                             01807
         prompt($"Operator: ");                                              01808
         *opstr* ← NULL;                                                     01809
         rdident($opstr, 0, 0);                                              01810
         %IF curchr # CA THEN getctc();%                                     01811
   initheq();                                                                01812
      %Control paameters%                                                    04162
         distribution ← TRUE;                                                04163
         collections ← mastacprintflag;                                      04164
         nic ← FALSE;                                                        04165
      %if wheel, set priority and bring down load average cuttoff%           01816
         IF (pcap ← enablw()) = -1 THEN BEGIN                                01817
            oljmlav ← 20400026;                                              01818
```

```
        END                                                              01819
      ELSE BEGIN                                                         01820
        disablw (pcap := -1);                                            01821
        setpriority(202B);                                               01822
        oljmlav ← 202700B6;                                              01823
        END;                                                             01824
%Type a warning message if this is the experimental system%              01825
  IF jdebug THEN                                                         01826
    BEGIN                                                                01827
    crlf();                                                              01828
    typeas($"Experimental Journal System");                              01829
    END;                                                                 01830
%Now parse control%                                                      01831
  % **** STATE ← jdlstate, spec; **** %                                  01832
  typech('$);                                                            01833
  IF %tnumber($jnum)% FALSE > 0 THEN                                     01834
    BEGIN                                                                01835
    IF curchr # CA THEN error();                                         01836
    crlf();                                                              01837
    printnum($jnum, distribution, collections, nic); %
    number--output specific document%                                    01838
    END                                                                  01839
  ELSE CASE IF curchr IN ['a, 'z] THEN curchr-40B ELSE curchr OF
                                                                         01840
    ='A:                                                                 01841
      BEGIN                                                              01842
      echo($"All");                                                      01843
      %getctc();%                                                        01844
      collections ← distribution ← nic ← TRUE;                           01845
      END;                                                               01846
    ='C: %Collection Copies%                                             01847
      BEGIN                                                              01848
      echo($"Collection Copies ");                                       01849
      collections ← answer();                                            01850
      END;                                                               01851
    ='D: %Distribution%                                                  01852
      BEGIN                                                              01853
      echo($"Distribution Copies ");                                     01854
      distribution ← answer();                                           01855
      END;                                                               01856
    ='G: %Go%                                                            01857
      BEGIN                                                              01858
      echo($"Go");                                                       01859
      %getctc();%                                                        01860
      crlf();                                                            01861
      IF NOT (distribution OR collections OR nic) THEN err($"No
      Parameters Set");                                                  01862
      rubabt ← 1;                                                        01863
      printall(distribution, collections, nic);                          01864
      END;                                                               01865
    ='H: %Hcdistfile collect%                                            01866
      BEGIN                                                              01867
      echo($"Hcdistfile garbage collect ");                              01868
      %getctc();%                                                        01869
      prntfg ← TRUE;                                                     01870
      jdquit();                                                          01871
```

```
    END;                                                        01872
='L: BEGIN                                                      01873
    mlavset();                                                  01874
    END;                                                        01875
='N: %Nic%                                                      01876
    BEGIN                                                       01877
    echo($"NIC Copies ");                                       01878
    nic ← answer();                                             01879
    END;                                                        01880
='O: %Output Processor Version%                                 01881
    BEGIN                                                       01882
    echo($"Output Processor Version ");                         01883
    CASE inpcuc() OF                                            01884
        ='E, ='X:                                               01885
            BEGIN                                               01886
            echo($"Experimental");                              01887
            %getctc();%                                         01888
            *oprocn* ← *xopname*;                               01889
            END                                                 01890
        = 'N:                                                   01891
            BEGIN                                               01892
            echo($"Normal");                                    01893
            %getctc();%                                         01894
            *oprocn* ← *opname*;                                01895
            END                                                 01896
        ='?:                                                    01897
            help($"Experimental",                               01898
                $"Normal",                                      01899
                0);                                             01900
        ENDCASE tqmarks();                                      01901
    END;                                                        01902
='P:                                                            01903
    BEGIN                                                       01904
    echo($"Printer File Name ");                                01905
    *lit* ← NULL;                                               01906
    txtlit($lit);                                               01907
    *ptstr* ← *lit*;                                            01908
    END;                                                        01909
='Q:                                                            01910
    BEGIN                                                       01911
    echo($"Quit ");                                             01912
    %getctc();%                                                 01913
    jdquit();                                                   01914
    reset();                                                    01915
    END;                                                        01916
='R: %reset parms%                                              01917
    BEGIN                                                       01918
    echo($"Reset Parameters");                                  01919
    %getctc();%                                                 01920
    distribution ← collections ← nic ← FALSE;                   01921
    END;                                                        01922
='S:                                                            01923
    BEGIN                                                       01924
    echo($"Status");                                            01925
    %getctc();%                                                 01926
    crlf();                                                     01927
```

```
        distcount ← hcstatus(:collcount, niccount);                    01928
        %type out distribution count%                                  01929
           crlf();                                                     01930
           typeas($"Distribution:  ");                                 01931
           *tempsr* ← STRING(distcount);                               01932
           typeas($tempsr);                                            01933
        %Type out Collection count%                                    01934
           crlf();                                                     01935
           typeas($"Collection:  ");                                   01936
           *tempsr* ← STRING(collcount);                               01937
           typeas($tempsr);                                            01938
        %Type out Nic count%                                           01939
           crlf();                                                     01940
           typeas($"Nic:  ");                                          01941
           *tempsr* ← STRING(niccount);                                01942
           typeas($tempsr);                                            01943
        %type out total%                                               01944
           crlf();                                                     01945
           typeas($"Total ");                                          01946
           *tempsr* ← STRING(distcount+collcount+niccount);            01947
           typeas($tempsr);                                            01948
        END;                                                           01949
     ='?:                                                              01950
        help($"Document Number",                                       01951
           $"All",                                                     01952
           $"Collection Copies ",                                      01953
           $"Distribuiton Copies ",                                    01954
           $"Go",                                                      01955
           $"NIC Copies ",                                             01956
           $"Output Processor Version ",                               01957
           $"Printer File Name ",                                      01958
           $"Quit",                                                    01959
           $"Reset Parameters",                                        01960
           $"Status",                                                  01961
           0);                                                         01962
   ENDCASE tqmarks();                                                  01963
   GOTO STATE;                                                         01964
END.                                                                   01965
```

L10 DATA

(MLK)L1ODATA (MLK)L1ODATA (MLK)L1ODATA (MLK)L1ODATA (MLK)L1ODATA
(MLK)L1ODATA (MLK)L1ODATA (MLK)L1ODATA (MLK)L1ODATA (MLK)L1ODATA

```
< NLS, L1ODATA.NLS;6, >, 11-OCT-74 10:43 KEV ;;;;
FILE l1Odata  %  L1O  <REL-NLS>L1ODATA  %% (L1O,)
(rel-nls,L1ODATA.rel,) %                                                    02
                                                                            03
REGISTER %here so DDT will use these on printout%                           04
   r1=1, r2=2, r3=3, r4=4, r5=5, p=7, wa=8, s=9, m=10, rp=11,               05
   a1=12, a2=13, a3=14, a4=15;                                              06
% *** GLOBAL DATA FOR PAGE 0 *** %                                         07
% Special stuff declared in page 0, loaded at 140B %                       037
   SET EXTERNAL                                                             039
      gstksz =  3072,  %length of general call stack%                       038
      pstksz = 20;%length of pattern stack%                                 08
%...KEEP THESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%                09
   DECLARE EXTERNAL %...important stuff...%                                 049
   %...user related data...%                                                050
      cinit,   % current initials of user %                                 010
   %...string analysis stuff...%                                            011
      flag = 1,        %the general flag%                                   012
      pstack[pstksz],     %pattern stack%                                   013
   %...editing global variables...%                                         014
      %...text editing global variables...%                                 034
         scndir,  % analyzer compiler scan direction %                      015
         rplsid,  % psid of statement being replaced in sc %                016
         sptr1[2], sptr2[2],  % for append T-string %                       017
         nsdbpt,  % byte pointer for statement being constructed%           041
   %...often used work areas...%                                            042
      swork, % string reading work area %                                   043
         swork1[6],                                                         044
      stcwrk, % string construction work area %                             045
         stcwr1[6],                                                         018
   %...set command and aptstr routine...%                                   053
      modeset,        %  option set  %                                     054
      modeshift,      %  case set    %                                     040
      mkrstay,        %true if do not move markers%                         019
      clmpty;   %reset value for clchng stack%                              020
% accessed in statement construction%                                       021
   DECLARE EXTERNAL STACK clchng[30];                                       022
% general call stack%                                                       023
   DECLARE EXTERNAL gstack[gstksz];                                         024
% DECLARE's %                                                              026
   DECLARE EXTERNAL STRING                                                  027
      sar[2000];        % DO NOT USE -used by string primitives%           028
   DECLARE EXTERNAL                                                         029
      %...general global variables...%                                      030
      sysgnl, %signal value%                                                031
      sysmsg, %signal message%                                              032
   %...other work areas...%                                                 033
      ps, pe; %cells for BETWEEN construct%                                 046
   %collector sorter%                                                       047
      DECLARE EXTERNAL                                                      048
         lngflg; %true if length is considered befor value in sort%         051
   DECLARE EXTERNAL STACK                                                   052
      btwstk[10];                                                           055
   %...spec stack...%                                                       060
      DECLARE EXTERNAL                                                      056
         spsk = spsk,    % spec stack pointer %
```

```
        spsk1= spsk,    % initial pointer %                              057
        b1[2], b2[2], b3[2], b4[2], b5[2],                              058
        spskt = b5;     % top of spec stack %                            059
    %...correspondence list...%  %PASSED%                                061
       DECLARE EXTERNAL                                                 066
        clhead[4],  %header, see clhdr record%                          062
           clistb[160], %80 two word entries, see clistr record %       063
           cliste, %end of clistb%                                      064
        clstad=clhead;  %address of clist header%                       065
FINISH of l1Odata                                                       035
```

(MLK)L1ORUNTIME (MLK)L1ORUNTIME (MLK)L1ORUNTIME (MLK)L1ORUNTIME (
(MLK)L1ORUNTIME (MLK)L1ORUNTIME (MLK)L1ORUNTIME (MLK)L1ORUNTIME (

```
< NLS. L1ORUNTIME.NLS;5, >, 11-OCT-74 12:59 DSM ;;;;
FILE l1oruntime  % L1O to <rel-nls>l1oruntime %% (L1O,) to (rel-nls,
L1ORUNTIME,) %                                                      02
% NOTES ABOUT RUNNING STANDALONE L1O PROGRAMS %                     01527
   % the following files should be loaded with your program to generate
   a standalone l1o program:                                        01528
      <rel-nls>l1odata.rel      - l1o data package                  01529
      <rel-nls>l1oruntime       - l1o runtime package               01530
      <system>tenex             - tenex jsys definitions            01531
   the following will be undefined unless they are provided by the
   user:                                                            01532
      sysovr                                                        01533
         this is a lable that is jumped to when a stack overflow occurs
                                                                    01534
         to see how NLS handles this see <nls, utilty, sysovr>      01535
                                                                    01536
      uflow
         this is a procedure that gets called when a stack underflow
         occurs                                                     01537
         to see how NLS handles this see <nls, auxcod, uflow>       01538
                                                                    01539
      err
         this procedure is called when some procedure detects an error
         condition                                                  01540
         to see how NLS handles this see <nls, utilty, err>         01541
                                                                    01542
      deferr
         this procedure is called when a signal has been propagated to
         the top of the stack                                       01543
         to see how NLS handles this see <nls, utilty, deferr>      01544
                                                                    01545
      filesc
         this procedure is called from procedure esc (end string
         construction) when esc detects that the strings being dealt
         with do not have the stastr bit set (i.e., they are not
         astrings)                                                  01546
         if a user is not dealing with any strings other than literal,
         local, or global strings (for example, strings within a data
         file), then this procedure need not be supplied.           01548
         to see how NLS handles this see <nls, utilty, filesc>      01547
                                                                    01549
      flcpfse
         this procedure is called from procedure cpfse when cpfse
         detects that the strings being dealt with do not have the
         stastr bit set (i.e., they are not astrings)               01550
         if a user is not dealing with any strings other than literal,
         local, or global strings (for example, strings within a data
         file), then this procedure need not be supplied.           01551
         to see how NLS handles this see <nls, utilty, flcpfse>     01552
                                                                    01553
      flfechcl
         this procedure is called from procedure fechcl when fechcl
         detects that the strings being dealt with do not have the
         stastr bit set (i.e., they are not astrings)               01554
         if a user is not dealing with any strings other than literal,
         local, or global strings (for example, strings within a data
         file), then this procedure need not be supplied.           01555
         to see how NLS handles this see <nls, utilty, flfechcl>    01556
   since l1o provides no way for specifying the starting point of a
   program, it is the user's responsibility to do this.  either of the
   following methods can be used to accomplish this:                01557
      after your load is done and before the program is saved, use the
```

```
    EXEC ENTVEC command                                                 01558
    after the load is done and before saving your program, go into
    ddt and start up your program at a label that you have supplied
    that will set up the entry vector and optionally do the save for
    you.  the instructions at this label cannot make use of any
    procedure local symbols and can not issue any procedure calls
    (including any implicit calls that might be dropped out by the
    compiler).  this is so because the PIO environment has not been
    established yet.  it is usually safe to accomplish this by using
    only assemoly language instructions.                                01559
when you are ready to actually run your program, it is necessary to
establish the L1O environment.  this is best accomplished by having
the first thing your program does is to execute the following
assembly language instruction:                                          01560
    !JSP 1,l1Oinit                                                      01561
no procedure calls (explicit or implicit) can be executed before the
L1O environment has been set up.  when your program gets control
back from l1Oinit, the environment will be established.                 01562
                                                                        01563
If you have problems or comments getting standalone L1O programs to
the point where the bugs are in your program and not associated with
the L1O environment, contact a systems programmer at SRI-ARC           01564
%                                                                       01565
% DECLARARIONTS %                                                        065
REGISTER                                                                053
    a1 = 12, %working register%                                         054
    a2 = 13, %working register%                                        01221
    a3 = 14, %working register%                                         055
    a4 = 15, %working register%                                        01222
    r1 = 1,  %working register%                                         056
    r2 = 2,  %working register%                                         057
    r3 = 3,  %working register%                                         058
    r4 = 4,  %working register%                                         059
    r5 = 5,  %working register%                                         060
    rp = 11, %record pointer%                                           061
    p = 7,   %pattern stack%                                            062
    wa = 8,   %pattern stack%                                          01223
    s = 9,   %general call stack pointer%                               063
    m = 10;  %mark stack pointer%                                       064
DECLARE EXTERNAL errmsbase = 140B;                                     01512
EXTERNAL l1Oinit;                                                       04
EXTERNAL                                                               0417
    readc;       %entry to READC routine%                              0418
DECLARE %pop transfer table% poptab=(baduuo,                           01324
    %1%  qcall, %no argument call pop%                                 01325
    %2%  qcall1, %single argument call pop%                            01326
    %3%  qcallm,  %up to 13 argument call pop%                         01327
    %4% qcallm2, %More than 13 argument call pop%                      01328
    %5-37B are not defined %                                           01329
        baduuo, baduuo, baduuo, baduuo, baduuo, baduuo,                01330
        baduuo, baduuo, baduuo, baduuo, baduuo, baduuo,                01331
        baduuo, baduuo, baduuo, baduuo, baduuo, baduuo,                01332
        baduuo, baduuo, baduuo, baduuo, baduuo¤@ baduuo,                01333
        baduuo, baduuo, baduuo, baduuo);                               01334
DECLARE EXTERNAL                                                       01246
    empty = 0, endfil = -1,                                            01247
```

```
    origin = 2 %=fbhdl%, nullch = 0, chbmty = 4407000000001B;          01248
DECLARE EXTERNAL                                                       01469
    sdbhdl=5;    %length of sdb header%                                01468
DECLARE EXTERNAL                                                       01510
    forward = 1;                                                       01511
% RECORDS %                                                           01437
(byteptr)   % field definitions for PDP-10 byte pointer word %         01520
    RECORD                                                             01521
    bpadr[18], bpindx[6], bpsize[6], bpbitpos[6];
                                                                       01522
(stidr) RECORD stpsid[18], stfile[8], stastr[1];                       01418
    % -- (filmnp, lodrng) and (filmnp, lodsdb) cheat and make use of
    the format of an stid -- so please don't change this without
    checking those routines and telling WHP %                          01419
    (stbw) RECORD stwc[9], stblk[9];                                   01420
    (stsdb) RECORD stpsdb[18];                                         01421
    (stast) RECORD stadr[18];                                          01422
    (stsidr) RECORD stsidf[6], stsid[30];
                                                                       01423
(clistr) RECORD %clist entry%                                          01424
    clst1[36],  %original or updated stid%                             01425
    clst2[36],  %successor stid%                                       01426
    clcc1[12],  %character count for first%                            01427
    clcc2[12],  %character count for second%                           01428
    clfixed[1]; %for aptstr%
                                                                       01429
    DECLARE EXTERNAL cll = 3, clmax = 50;
                                                                       01430
(clhdr) RECORD %clist header%                                          01431
    clcnt[9],   %count of entries in use%                              01432
    clfno1[7],  %type used to generate clist%                          01433
    clfno2[7],  %type used to generate clist%                          01434
    cltype[6],  %type used to generate clist%                          01435
    clbuff[18]; %address of buffer%
                                                                       01436
(ring) RECORD %ring1 is length%                                        01438
    rsub[18],     %psid of sub of this statment%                       01439
    rsuc[18],     %psid of suc of this statement%                      01440
    rsdb[18],     %psdb of sdb for this statement%                     01441
    rinst1[7],  %DEX interpolation string-- scratch space%             01442
    rinst2[7],  %DEX interpolation string-- scratch space%             01443
    rdummy[1],    %DEX dummy flag-- scratch space%                     01444
    repet[3],     %DEX repetition-- scratch space%                     01445
    rhf[1],       %head flag. true if this is head of plex%            01446
    rtf[1],       %tail flag, true if tail of plex%                    01447
    rnamef[1],    %name flag, true if statement has a name%            01448
    rnull[2],     %unused%                                             01449
    rnameh[30],   %name hash for this statement%                       01450
    rsid[30];     %statement identifier%                               01451
    %although only need four words, use five so that have room         01452
    to grow%
                                                                       01453
(sdbhead) RECORD %sdbhdl is length%                                    01454
    sgarb[1],     %true if this sdb is garbage%                        01455
    slength[9],   %number of words in this sdb%                        01456
    schars[11],   %number of characters in this statement%             01457
```

```
    slnmdl[7],    %left name delimiter for statement%                    01458
    srnmdl[7],    %right name delimiter for statement%                   01459
    spsid[18],    %psid of the statement for this sdb%                   01460
    sname[11],    %position of character after name%                     01461
    stime[36],    %date and time when this sdb created%                  01462
    sinit[21];    %initials of user who created this sdb%                01463
    %sgarb and slength must be in the first word of the header           01464
    for newsdb% %although only need four words, use five so              01465
    that have room to grow%                                              01466
                                                                         01467

(xllOinit) PROCEDURE; %stack overflow/under flow code%                      05
  %accessed by a JSP r1, llOinit instruction%                               06
  % initialize the pop transfer instruction%                                07
     (llOinit): a1 ← krummy; !MOVEM a1,41B;                                 08
  % initialize stack pointers%                                              09
     %general call stack%                                                  010
        s ← -$gstksz; !HRL s,s; !HRRI s,gstack;                            011
        m ← s;                                                             012
        gstack ← $uflow;                                                   013
     %pattern stack%                                                       014
        p ← -$pstksz; !HRL p,p; !HRRI p,pstack;                            015
     % spec stack %                                                        016
        spsk ← spsk1;                                                      017
  % reset clist change stack %                                            018
     RESET clchng; % before any string constructions are done%             019
     clmpty ← clchng;                                                      020
     GOTO [r1];                                                            023
  % instruction for pop transfer%                                         021
     (krummy): !JSP r5,ppaprt;                                             022
  END.                                                                     051
  %%                                                                      066
```

```
% call and pop routines%                                                  067
  EXTERNAL ppaprt, qcall, qcalll, qcallm;                                0162
  (sysrtn) PROCEDURE; %for returns%                                      068
    %get here by a JRST sysrtn%                                          069
    !MOVE s,m;  %back up the stack%                                      070
    !POP s,a4;  %save the return location%                               071
    !POP s,m;  %restore the mark%                                        072
    !JRST (a4) END.
                                                                         073
  (sys2rt) PROCEDURE; %return with 2 results%                            074
    !MOVE s,m;                                                           075
    !POP s,a4;                                                           076
    !POP s,m;                                                            077
    !MOVEM a2,3(s);                                                      078
    !JRST (a4) END.
                                                                         079
  (sysmrt) PROCEDURE; %for returns with multiple results%                080
    % register a2 contains number-1 of multiple results%                081
    !MOVN a3,a2;                                                         082
    !ADD a3,s;                                                           083
    !HRL a3,a3; %a3 left points to first on stack%                       084
    !HRR a3,m;                                                           085
    !AOJ a3,; %a3 right points to destination%                           086
    !HRRZ a4,a3;                                                         087
    !ADD a4,a2; %a4 points to last loc moved to%                         088
    !BLT a3,@a4;                                                         089
    !JRST sysrtn END.
                                                                         090
  (syssig)PROC;                                                          091
    WHILE [[m-1]].LH = 0 DO                                              092
      IF (m .A 18M) <= ([m-1] .A 18M)  THEN                              093
        GOTO sysovr                                                      094
      ELSE                                                               095
        IF  [m←[m-1]] = Suflow THEN                                      096
          deferr();                                                      097
    %by here, m points to mark beyond one with signal loc%               098
    [m] ← [[m-1]].LH; %change return loc to signal code%                 099
    RETURN;                                                              0100
    END.
                                                                         0101
  (repsig)PROC(value);                                                   0102
    %Repeat a signal with value as sysgnl%                               0103
    SIGNAL(value);                                                       0104
    END.                                                                 0105
  (callsig)PROC(value, message);                                         0106
    %generate the specified signal with the spcified message.  this
    routine is useful if the calling procedure wishes a chance to
    catch the signal itself -- would not be possible if the procedure
    generated the signal itself.%                                        0107
    SIGNAL(value, message);                                              0108
    END.                                                                 0109
  DECLARE FIELD popldb=[40B, 9:27], l40ac=[40B, 4:23];                   0110
  (cllp) PROCEDURE; %dummy procedure for calls%                          0111
    (ppaprt):                                                            0112
      %get here by a JSP r5,ppaprt in location 41B%                      0113
      % fast code to get UUO number: %                                   0114
```

```
        !HLRZ r4,40B;                                                   0115
        !ROT r4,-11B;                                                   0116
        !MOVEI r5,0(r5); % left half zero %                             0117
        !JRST @poptab(r4); % goto UUO %                                 0118
      (baduuo):  % illegal UUO (see poptab) %                           0119
      r1 ← $"Illegal UUO at " + 1;                                      0120
      !HRLI r1,7B2;                                                     0121
      !JSYS psout;                                                      0122
      r1 ← 101B;                                                        0123
      r2 ← (r5 - 1) .A 18M;                                             0124
      r3 ← 8;                                                           0125
      !JSYS nout;                                                       0126
      !JSYS haltf;                                                      0127
      !JSYS haltf;                                                      0128
    (qcall):                                                            0129
      %get here by a CALL pop%                                          0130
      !PUSH s,m;  %save the mark%                                       0131
      !PUSH s,r5;  %save the return location%                           0132
      !MOVE m,s;  %new mark%                                            0133
      !JRST @40B;                                                       0134
    (qcall1):                                                           0135
      %get here by a CALL1 pop%                                         0136
      !PUSH s,m;  %save the mark%                                       0137
      !PUSH s,r5; %save the return location%                            0138
      !MOVE m,s;  %new mark%                                            0139
      !MOVEM a1,1(m); %store the single argument%                       0140
      !JRST @40B;                                                       0141
    (qcallm):                                                           0142
      %get here by a CALLM pop%                                         0143
      !ROT r4,4;  % get AC field in right end %                         0144
      !ANDI r4,17B; % only 4 bits %                                     0145
      !HRLI r4,0(r4); % copy into left half %                           0146
      !SUB s,r4; %now same as before started call%                      0147
      !PUSH s,m;  %save the mark%                                       0148
      !PUSH s,r5; %save the return location%                            0149
      !MOVE m,s;  %new mark%                                            0150
      !JRST @40B;                                                       0151
    (qcallm2):                                                          0152
      %get here by a CALLM pop%                                         0153
      !ROT r4,4; % get AC field in right end %                          0154
      !ANDI r4,17B; % only 4 bits %                                     0155
      !SUB s,0(r4); %now same as before started call%                   0156
      !PUSH s,m;  %save the mark%                                       0157
      !PUSH s,r5; %save the return location%                            0158
      !MOVE m,s;  %new mark%                                            0159
      !JRST @40B;                                                       0160
    END.                                                                0161
%%                                                                      0163
```

```
% String Construction%                                                  0164
   (bsc) %Begin string construction.  Called with STID (or pointer to
   string with stastr TRUE) of the destination.%
                                                                        0165
                                                                        0166
      PROCEDURE (dest);                                                 0167
      rplsid ← dest;                                                    0168
      %set up byte pointer and SAR for string construction%             0169
      nsdbpt ← chbmty + $sar;                                           0170
      sar.L ← empty;                                                    0171
      %reset stack for modified clist entries%                          0172
      clchng ← clmpty;
      RETURN END.                                                       0173

   (esc) %End string construction.%
                                                                        0174
      PROCEDURE;                                                        0175
      IF rplsid.stastr THEN %A-string%                                  0193
         BEGIN                                                          0194
         */rplsid.stadr/* ← *sar*;                                      0195
         RETURN;                                                        0196
         END;                                                           0197
      filesc();                                                        01523
      RETURN;                                                          01524
      END.                                                              0275

   (kps) %argument points to string to be appended to the statement
   being constructed%
                                                                        0276
      PROCEDURE (asfrom);                                               0277
      REF asfrom;                                                       0278
      LOCAL bptr; %byte pointer for reading characters%                 0279
      IF (sar.L ← sar.L + asfrom.L) > sar.M THEN                        0280
         err($"NLS internal error:  string too long");                  0281
      bptr ← chbmty + &asfrom;                                          0282
      a2 ← asfrom.L;                                                    0283
      a3 ← nsdbpt;                                                      0284
      a4 ← bptr;                                                        0285
      UNTIL (a2 ← a2-1) < 0 DO ↑a3 ← a1 ← ↑a4;                          0286
      nsdbpt ← a3;                                                      0287
      RETURN END.                                                       0288

   (apachr) %passed character as argument.  appends to the statement
   being constructed in sar.%
                                                                        0289
      PROCEDURE (ch);                                                   0290
      IF (sar.L ← sar.L+1) > sar.M THEN err($"NLS internal error:
      string too long");                                                0291
      ↑nsdbpt ← ch;                                                     0292
      RETURN END.                                                       0293

   (aptstr) %Append tstring.  Argument is pointer to two T-pointers
   which delimit the substring to be appended to the statement
   currently being constructed.%
                                                                        0294
      PROCEDURE (tp);                                                   0295
      LOCAL                                                             0296
```

```
    cl,         %location of record for a cl to be updated%
    end,        %location of end of cl table%
    cltab,      %location of cl table%
    flhd,       %file header loc%
    ch,         %character read from the tstring%
    bfrom,       %source byte pointer%
    lenstat,    %number of chars from string%
    lenblank; %number of chars from string%
REF cl;
IF [tp] # [tp+2] OR [tp+1] <= empty THEN
    BEGIN
    modeset ← 0; %reset global modeset flag %
    err($"illegal string designation");
    END;
IF [tp+1] >= [tp+3] THEN
    BEGIN
    modeset ← 0; %reset global modeset flag %
    RETURN;
    END;
ON SIGNAL ELSE modeset ← 0;
stcwrk ← [tp];
stcwrl ← [tp+1];
fechcl (forward, $stcwrk);
% stcwrk[2] now contains length + 1 of source string %
% calculate number of chars to take from source string %
CASE stcwrk[2] OF
    <= [tp+1] :
        BEGIN
        lenstat ← 0;
        lenblank ← [tp+3] - [tp+1];
        END;
    < [tp+3] :
        BEGIN
        lenstat ← stcwrk[2] - [tp+1];
        lenblank ← [tp+3] - stcwrk[2];
        END;
    ENDCASE
        BEGIN
        lenstat ← [tp+3] - [tp+1];
        lenblank ← 0;
        END;
&cl ← cltab ← [clstad].clbuff;
    %address of start of clist%
end ← &cl + [clstad].clcnt * cll;
    %end of clist%
ON SIGNAL ELSE;
IF modeset THEN
    BEGIN % mode set now in operation %
    modeset ← 0; %reset the global flag%
    UNTIL (lenstat ← lenstat-1) < 0 DO
        BEGIN
        ch ← READC($stcwrk);
        %now append the character%
        CASE modeshift OF %check for forced case%
            =0: apachr(IF ch IN ['a,'z] THEN ch-40B ELSE ch);
                %forced upper case%
```

```
            =1: apachr(IF ch IN ['A,'Z] THEN ch+40B ELSE ch);
                %forced lower case%
            ENDCASE apachr(ch); %no case change%
         END;
      END
   ELSE %no mode change. just copy across%
      BEGIN
      IF NOT mkrstay THEN
         IF NOT rplsid.stastr THEN
            BEGIN
            IF rplsid.stfile = [tp].stfile THEN
               UNTIL &cl = end DO
                  BEGIN
                  IF cl.clstl = [tp] AND
                     cl.clccl IN [[tp+1],[tp+3]) AND
                     NOT cl.clfixed THEN
                        BEGIN
                        cl.clfixed ← TRUE;
                        cl.clccl ←
                           sar.L + 1 + cl.clccl - [tp+1];
                        PUSH &cl ON clchng;
                        END;
                  &cl ← &cl + cll;
                  END
            ELSE cldsr(tp);
            END;
      IF (sar.L ← sar.L + lenstat) > sar.M THEN
         err(@"NLS internal error:  string too long");
      a4 ← stcwrk[4];  %byte pointer made by fechcl%
      a2 ← lenstat;
      a3 ← nsdbpt;
      UNTIL (a2 ← a2-1) < 0 DO ↑a3 ← a1 ← ↑a4;
      nsdbpt ← a3;
      END;
   UNTIL (lenblank ← lenblank-1) < 0 DO apachr(SP);
   mkrstay ← FALSE;
   RETURN END.
```

```
(cldsr) %Called with address of two tpointers.  For all clist
entries, if between these two pointers, then set to "deleted".%

   PROCEDURE (tp);
   LOCAL end, cl;
   REF cl;
   &cl ← [clstad].clbuff;
      %address of start of clist%
   end ← &cl + [clstad].clcnt * cll;
      %end of clist%
   UNTIL &cl >= end DO
      BEGIN
      IF cl.clstl = [tp] AND
         cl.clccl IN [[tp+1],[tp+3]) AND
         NOT cl.clfixed THEN
         BEGIN
         cl.clst2 ← cl.clstl := endfil;
         cl.clcc2 ← [tp+1];
```

```
            END;                                                    01506
         &c1 ← &c1 + c11;                                           01507
         END;                                                       01508
      RETURN END.                                                   01509

% Reading characters from SDB%                                      0421
   (fechc1)                                                         0422
      %Documentation%                                               0423
         %This routine is called to initialize a work area for reading
         characters from a statement.  Arguments are: direction of
         reading characters (=0 then backwards) and the address of the
         7 word work area (of which the last 2 words are no longer
         used).                                                     0424
         If characters are to be read from a statement then         0425
            when calling FECHC1, the first two cells of the work area
            must contain a Tpointer.  A character count of one
            indicates the first character of the statement. FECHC1 will
            initialize the rest of the work area. The first word of the
            word area always has the PSID of the statement.  The second
            word has the character count.  The third word contains a
            bound on characters to be read.  ENDCHR's are returned
            after reach this bound.  The fourth word has the direction
            of readout for use by readc.  A READC(x) actually results
            in the value of x being loaded into register wa followed by
            a JSP a4,readc.                                         0426
            The fifth word of the work area contains a byte pointer to
            the character last read from the statement. Thus an ILDB
            instruction may be used to get the next character if the
            direction is forward.  If the direction of reading is
            backward, then the byte pointer is decremented by in-line
            code.                                                   0427
            To read characters from the statement execute a READC(x)
            where the value of x is the address of the work area to be
            used.  The character is returned as the value of the READC.
             Subsequent READC's will return the following characters.
                                                                    0428
            To change position or direction within the statement the
            work area must be reinitialized by calling FECHC1 again, as
            described above.  There may however be more than one work
            area currently in use, and these may be changed
            independently.                                          0429
         If characters are to be read from an A-string then         0430
            the first word of the work area contains the address of the
            A-string instead of a PSID.  The second word is 1 if the
            first character of the string is to be read next, two if
            the second, etc.  Characters may be read out of an A-string
            in either direction, just like a statment.  Endcharacters
            are returned when the string is exhausted.%             0431
      PROCEDURE (dir, worka);                                       0432
      LOCAL                                                         0433
         wp,    %word postion of the starting character%           0434
         cp,    %character postion of the starting character%      0435
         stdb,  %stdb for statement%                               0436
         rng,   %location of ring element%                         0437
         addr; %address of the word containing starting character% 0438
      REF rng;                                                      0439
```

```
   %set work+2 to bound%                                                0440
   IF [worka] = endfil THEN %set to null string%                        0441
      BEGIN                                                             0442
      [worka+3] ← 2; %readc will return ENDCHR%                         0443
      RETURN;                                                           0444
      END                                                               0445
   ELSE IF [worka].stastr THEN %A-string%                               0446
      BEGIN                                                             0447
      addr ← [worka].stadr;                                             0448
      [worka+2] ← IF pe THEN pe ELSE [addr].L + 1;                      0449
      addr ← addr + 1;                                                  0450
      END                                                               0451
   ELSE %SDB%                                                           0452
      addr ← flfechcl( dir, worka : stdb );                            01526
   %find word and character position%                                   0462
   IF dir THEN %scan forward%                                           0463
      BEGIN                                                             0464
      [worka+3] ← 1; %direction%                                        0465
      END                                                               0466
   ELSE %scan backwards%                                                0467
      BEGIN                                                             0468
      [worka+2] ← IF ps THEN ps ELSE 1; %change bound%                  0469
      [worka+3] ← 0; %direction%                                        0470
      END;                                                              0471
   %make byte pointer to the previous (forward) or current
   (backward) character%                                                0472
   DIV ([worka+1]-[worka+3])/5, wp, cp;                                 0473
   [worka+4] ← 440700B6 - cp * 70000B6 + addr + wp;                     0474
   RETURN END.                                                          0475

(openswork) PROCEDURE;                                                  0476
   fechcl(1, @swork);                                                   0477
   RETURN END.                                                          0478

(savpos) PROCEDURE;                                                     0479
   PUSH ps ON btwstk;                                                   0480
   PUSH pe ON btwstk;                                                   0481
   RETURN END.                                                          0482

(respos) PROCEDURE;                                                     0483
   POP btwstk TO pe;                                                    0484
   POP btwstk TO ps;                                                    0485
   RETURN END.                                                          0486

(xxreadc) PROCEDURE; %This is the routine that is called to read a
character by the READC construct.  Code to call is JSP a4,readc so
return loc is in A4.%                                                   0487
   (readc):                                                             0488
   %cant have locals since is not called by usual procedure linkage.
    wa is a register containing the address of the work area.%          0489
   !SKIPG a1,3(wa);                                                     0490
      !JRST readc1;                                                     0491
   !SOJN a1,readend;                                                    0492
   %forward scan%                                                       0493
      !AOS a1,1(wa);                                                    0494
      !CAMLE a1,2(wa);                                                  0495
```

```
        !JRST rdend1;                                                   0496
      !ILDB a1,4(wa); %get the character%                               0497
      !JRST 0(a4);                                                      0498
    (readc1):                                                           0499
    !JUMPN a1,readend;                                                  0500



    %backward scan%                                                     0501
      !SOS a1,1(wa);                                                    0502
      !CAMGE a1,2(wa);                                                  0503
        !JRST rdend2;                                                   0504
      %back up byte pointer%                                            0505
      !MOVE a1,4(wa);                                                   0506
      !ADD a1,=7B10;                                                    0507
      !CAIG a1,0;                                                       0508
      !SUB a1,=430000000001B;                                           0509
      !MOVEM a1,4(wa);                                                  0510
      !LDB a1,a1; %get the character%                                   0511
      !JRST 0(a4);                                                      0512
    (rdend2): !AOSA 1(wa); %adjust for backward overrun, NOTE SKIP%     0513

    (rdend1): !SOS 1(wa); %adjust for forward overrun, SKIPPED OVER%    0514

    (readend): %out of bounds%                                          0515
      a1 ← ENDCHR;                                                      0516
      !JRST 0(a4);                                                      0517
    END.                                                                0518

% A-string routines%                                                   0547
  (asrref) %create a pointer to astring with stastr field set.%         0548

    PROCEDURE (ast);                                                    0549
    ast.stastr ← TRUE;                                                  0550
    RETURN (ast) END.                                                   0551

  DECLARE % byte pointers for chbptr %                                  0552
    chparray=(                                                          0553
      350700000001B,                                                    0554
      260700000001B,                                                    0555
      170700000001B,                                                    0556
      100700000001B,                                                    0557
      070000000001B);                                                   0558

  DECLARE cshift=(7,14,777761B %-15%, 777770B %-8%, 777777B %-1%);      0559
  EXTERNAL ldchr, chbptr, apchr;                                        0560
  DECLARE ascomm=(0,774B9,77776B7,7777777B5,-400B);                     0561
  (xutilty) PROCEDURE;                                                  0562
    % a place for several "fast" routines to reside.  NOTE: These
    routines cannot use locals or make calls since they are lables:
    they avoid the stack setup instructions at procedure entry.  They
    are called, however, as procedures with arguments. %                0563
    (ldchr):  % (astr, charno) %                                        0564
      % return a specified character from an astring %                  0565
      !MOVE a3,@1(m);  % string max,,len %                              0566
      !MOVE a1,2(m);  % char number %                                   0567
      !CAILE a1,0(a3); % over end? %                                    0568
      !JRST ldchr1;  % yes, return ENDCHR %                            0569
      !JUMPE a1,sysrtn; % zeroth char - return zero %                   0570
```

```
      !MOVEI a1,-1(a1);                                                 0571
      !IDIVI a1,5;  % char-1/5 %                                        0572
      !ADD a1,1(m); % add address of string %                           0573
      !MOVE a1,1(a1);  % get word from string %                         0574
      !ROT a1,@cshift(a2);  % rotate and mask %                         0575
      !ANDI a1,1775;  % faster than LDB %                               0576
      RETURN;                                                           0577
      (ldchr1):                                                         0578
         RETURN(ENDCHR);                                                0579
   (chbptr):  % (char) %                                                0580
      % return a partial byte pointer given a character number.  The
      resulting byte pointer needs to have the address of the
      astring added to it %                                             0581
      !SKIPN a1,1(m);  %check for zero char count %                     0582
      !JRST chbpt1;                                                     0583
      !MOVEI a1,-1(a1);  %subtract one %                                0584
      !IDIVI a1,5;                                                      0585
      !ADD a1,chparray(a2);  %get partial pointer %                     0586
      RETURN;                                                           0587
      (chbpt1):                                                         0588
         RETURN(cnbmty);                                                0589
   (apchr): % (char, astr) %                                            0590
      % Append the given character to the specified astring.  If the
      string is at it's max length an exceed capacity error results
      %                                                                 0591
      !MOVE a2,@2(m);  % get max,,len from string %                     0592
      !MOVSI a1,1(a2); % get len+1 %                                    0593
      !CAMLE a1,a2;  % over max ? %                                     0594
      !JRST apxced;                                                     0595
      !HLRM a1,@2(m);  % store len+1 %                                  0596
      !MOVEI a2,0(a2);  % isolate new length-1 = old len %              0597
      (apstore):                                                        0598
         !IDIVI a2,5;                                                   0599
         !ADD a2,chparray(a3);  % get partial pointer %                 0600
         !ADD a2,2(m);  % plus string address %                         0601
         !MOVE a1,1(m);  % get character %                              0602
         !DPB a1,a2;  % and store it %                                  0603
         RETURN;                                                        0604
      (apxced):                                                         0605
         err(@"NLS internal error:  string too long");                  0606
   NULL END.                                                            0607
(stbptr) %character byte pointer%                                       01513
   PROCEDURE (charno);                                                  01514
   %creates a partial byte pointer to the character determined by
   the character number passed as argument.  The resulting byte
   pointer needs to have the address of the string (NOT ASTRING)
   added to it.%                                                        01515
   LOCAL wp, cp;                                                        01516
   IF charno = empty THEN RETURN (440700000000B);                       01517
   DI  (charno-1) / 5, wp, cp;                                          01518
   RETURN (chparray[cp] + wp - 1) END.
                                                                        01519
(ascom) PROCEDURE(astr1, astr2, relation);                              0608
   % accepts addresses of 2 a-strings.  returns logical value of
   astr1 <relation> astr2 %                                             0609
   LOCAL reltabl;                                                       0610
```

```
r1 ← -1;  % set flag in r1 - means must do all of compare %
r2 ← relation;
(comstr):  % called by ascom and compas %
   % r1 is a flag: =0 means return false if a mismaxch (compas
   call). <0 means must compare entire string and return reltab
   (ascom call). >0 means return the sign of astr1 = astr2
   (cmpstr call) %
   IF [astr1].L # [astr2].L THEN BEGIN
      !JUMPE r1,ascom6;  % RETURN(FALSE) if called from compas %

      !SKIPE lngflg;  % cmpstr: IF lngflg AND lengths not equal %

      !JUMPG r1,ascom5;  % cmpstr call - compare lengths %
      !CAIN r2,2;  % is relation = ? %
      !JRST ascom6;  % yes.  RETURN(FALSE) %
      !CAIN r2,6;  % if relation # ? %
      !JRST ascom7  % yes.  RETURN(TRUE) %
      END;
   !IMULI r2,3;  % r2 ← r2*3+1 %
   !MOVEI r2,1(r2);  % plus one %
   a3 ← [m+1]; % $astr1 %
   a4 ← [m+2];  % $astr2 %
   !HRRZ a1,0(a3);  % len of astr1 %
   !HRRZ a2,0(a4);  % len of astr2 %
   !CAIG a2,0(a1);  % find min len %
   !MOVEI a1,0(a2);  % now a1 has the min %
   !IDIVI a1,5;  % length/5: a1=number of words, a2=mask index %

   (ascom2):
      !MOVEI a3,1(a3);  % add one to pointer %
      !MOVEI a4,1(a4);  % ditto %
      !MOVE r3,0(a3);  % get a word of string %
      !TRZ r3,1;  % turn off low order bit %
      !SOJL a1,ascom1;  % last word of str#ng ? %
      !MOVE r4,0(a4);  % get other word %
      !TRZ r4,1;  % and turn of low order bit %
      !SUB r3,r4;  % now, take difference %
      !JUMPE r3,ascom2;  % if equal, keep going %
      !JUMPE r1,ascom6;  % quit here if we can (i.e. from compas)
      %
   (ascom3):
      !TLNE r3,4B5;  % compute sign of result:  sign bit on? %

      !SKIPA r3,=-1;  % yes, get -1 %
      !MOVEI r3,1;  % no, get one %
   (ascom4):
      !JUMPG r1,ascom8;  % cmpstr call - return sign, not reltab
      %
      !ADDI r3,0(r2);  % add in reltab index %
      RETURN(reltab[r3]);  % return result from table %
   (ascom1):
      !JUMPE a2,ascom5;  % last word of str.  any chars in it? %

      !AND r3,ascomm(a2);  % and off extraneous chars %
      !MOVE r4,0(a4);  % get word from other string %
      !AND r4,ascomm(a2);  % and it also %
```

```
      !SUB r3,r4;  % take difference %                                    0657
      !JUMPN r3,ascom3;  % check sign if not equal %                     0658
                                                                          0659
   (ascom5):
      !HRRZ r3,@1(m);  %last words equal.  compare lengths %             0660
      !HRRZ r4,@2(m);  % get other length %                              0661
      !SUBI r3,0(r4);  % take difference %                               0662
      !JUMPE r3,ascom4;  % equal, return result %                        0663
      !JRST ascom3;  % not equal, compute sign %                         0664
                                                                          0665
   (ascom6):
      RETURN(FALSE);                                                      0666
                                                                          0667
   (ascom7):
      RETURN(TRUE);                                                       0668
                                                                          0669
   (ascom8):
      RETURN(r3); % return sign for cmpstr call %                         0670
                                                                          0671
NULL END.
(compas) PROCEDURE(astr1, astr2);                                         0672
   % compares the contents of two a-strings.  Returns true if the
   contents match, false otherwise %                                      0673
   r1 ← 0; r2 ← 2; % relation = true %                                    0674
   GOTO comstr END.                                                       0675
(cmpstr) PROCEDURE(astr1, astr2);                                         0676
   % return -1 if astr1 < astr2, 0 if equal, 1 if astr1 > astr2 %
                                                                          0677
   r1 ← 1; % set flag >0 means return sign value %                        0678
   GOTO comstr END. % value of r2 does not matter %                       0679
(repchr) %replace a character in an astring%
                                                                          0687
   PROCEDURE (char, ast, chrno);                                          0688
   %------------------%                                                  0689
   LOCAL cnt;                                                             0690
   REF ast;                                                               0691
   CASE chrno OF                                                          0692
      > ast.M : err($"NLS internal error:  string too long");
      %string overflow%                                                   0693
      > ast.L : % blank fill if necessary %                               0694
         BEGIN                                                            0695
         IF (cnt ← chrno - ast.L) > 1 THEN apblnk(&ast, cnt-1);           0696
         ast.L ← chrno;                                                   0697
         END;                                                             0698
      <=0: err($"repchr called with chrno<=0");                           0699
      ENDCASE;                                                            0700
   !MOVE a2,3(m); % the char number %                                     0701
   !MOVEI a2,-1(a2);  % minus one %                                       0702
   !JRST apstore                                                          0703
   END.
                                                                          0704
(apblnk) PROCEDURE(ast, cnt);                                             0705
   % Append cnt blanks to the designated astring. %                       0706
   REF ast;                                                               0707
   FOR cnt DOWN UNTIL < 1 DO *ast* ← *ast*, SP;                           0708
   RETURN END.
                                                                          0709
(apsr) %The purpose of this routine is to append one A-string onto
another.  Arguments are two A-string addresses.  The first string is
appended to the second.  An ERROR (NLS internal error:  string too
long) is generated if the maximum length of the latter string is too
```

```
short to contain the result.%                                          0710
   PROCEDURE (asfrom, asto);                                           0711
   LOCAL                                                               0712
      wp,     % word position %                                        0713
      cp,     % character position %                                   0714
      bfrom, % byte pointer in source string %                         0715
      bto;    % byte pointer in destination string %                   0716
   REF asfrom, asto;                                                   0717
   bfrom ← chbmty + &asfrom;                                           0718
   bto ← chbptr(asto.L) + &asto;                                       0719
   IF (asto.L ← asto.L + asfrom.L) > asto.M THEN                       0720
      err($"NLS internal error:  string too long");                    0721
   a2 ← asfrom.L;                                                      0722
   UNTIL (a2 ← a2-1) < 0 DO ↑bto ← a1 %use a1% ← ↑bfrom;               0723
   RETURN END.                                                         0724
(cpysr) %To copy one A-string into another, call this routine with
the address of the source string and the address of thedestination
string (in that order).  An ERROR is generated if the maximum length
of the destination string is shorterthan the source string.%           0725
   PROCEDURE (asfrom, asto);                                           0726
   REF asfrom, asto;                                                   0727
   CASE asfrom.L OF                                                    0728
      > asto.M : err($"NLS internal error:  string too long");         0729
      <= empty : asto.L ← empty;                                       0730
      ENDCASE                                                          0731
         BEGIN                                                         0732
         asto.L ← asfrom.L;                                            0733
         mvbfbf(&asfrom+1, &asto+1, (asfrom.L+4)/5);                   0734
         END;                                                          0735
   RETURN END.                                                         0736
(chpair) %extract substring and append to another string%              0737
   PROCEDURE(ast1, lower, upper, ast2);                                0738
   LOCAL bto, bfrom, lenstr, lenblank;                                 0739
   REF ast1, ast2;                                                     0740
   IF upper < lower THEN RETURN;                                       0741
   lower ← MAX(1, lower);                                              0742
   CASE ast1.L OF                                                      0743
      < lower :                                                        0744
         BEGIN                                                         0745
         lenstr ← 0;                                                   0746
         lenblank ← upper-lower+1;                                     0747
         END;                                                          0748
      < upper :                                                        0749
         BEGIN                                                         0750
         lenstr ← ast1.L-lower+1;                                      0751
         lenblank ← upper-ast1.L;                                      0752
         END;                                                          0753
      ENDCASE                                                          0754
         BEGIN                                                         0755
         lenstr ← upper-lower+1;                                       0756
         lenblank ← 0;                                                 0757
         END;                                                          0758
```

```
   IF ast2.L + lenstr + lenblank > ast2.M THEN                          0759
      err($"NLS internal error:  string too long");                     0760
   bto ← chbptr(ast2.L) + &ast2;                                        0761
   bfrom ← chbptr(lower-1) + &ast1;                                     0762
   ast2.L ← ast2.L + lenstr + lenblank;                                 0763
   a2 ← lenstr;                                                         0764
   a3 ← bto;                                                            0765
   a4 ← bfrom;                                                          0766
   UNTIL (a2 ← a2 - 1) < 0 DO ↑a3 ← a1 ← ↑a4;                           0767
   a2 ← lenblank;                                                       0768
   a1 ← SP;                                                             0769
   UNTIL (a2 ← a2 - 1) < 0 DO ↑a3 ← a1;                                 0770
   RETURN END.
                                                                        0771
```

(mvbfbf) %This routine moves a buffer of words. Arguments are address of source, address of destination, and number of words to transfer. It returns the address of the last word into which data was moved.%

```
                                                                        01224
   PROCEDURE (bfrom, bto, nw);                                          01225
   LOCAL lw;                                                            01226
   IF nw = 0 THEN RETURN;                                               01227
   lw ← nw + bto - 1; %last word to be transferred to%                  01228
   IF bto IN (bfrom, bfrom+nw) THEN                                     01229
      BEGIN                                                             01230
      a1 ← bfrom;                                                       01231
      a2 ← nw; a3 ← a2;                                                 01232
      a2 ← a2 + a1; %pointer into source%                               01233
      a3 ← a3 + bto; %pointer into destination%                         01234
      UNTIL (a2 ← a2-1) < a1 DO                                         01235
         BEGIN                                                          01236
         a3 ← a3 - 1;                                                   01237
         [a3] ← [a2];                                                   01238
         END;                                                           01239
      END                                                               01240
   ELSE %can use block transfer instruction%                            01241
      BEGIN                                                             01242
      !HRL a1,bfrom; !HRR a1,bto; !BLT a1,@lw                           01243
      END;                                                              01244
   RETURN (lw) END.
                                                                        01245
```

```
DECLARE hshmsk=(                                                        01318
   774B9,                                                               01319
   77776B7,                                                             01320
   7777777B5,                                                           01321
   777777777 4B2,                                                       01322
   77777777776B);
                                                                        01323
```

(hash) %An a-string address as argument. The hash code for that string is returned.%

```
                                                                        01299
   PROCEDURE (ast);                                                     01300
   LOCAL nw, l, gen1, gen2, old1, cp;                                   01301
   REF ast;                                                             01302
   IF (old1 ← ast.L) <= empty THEN RETURN(0);                           01303
   DIV (old1 + 4)/5, nw, cp;                                            01304
```

```
   FOR i ← 1 UP UNTIL = nw DO ast[i] ← ast[i] .A hshmsk[4];
   ast[nw] ← ast[nw] .A hshmsk[cp];
   gen1 ← 0;
   gen2 ← 1;
   UNTIL nw = 0 DO
      BEGIN
      gen1 ← ast[nw] * gen2 + gen1 / 17;
      gen2 ← gen2 * 43;
      nw ← nw - 1;
      END;
   al ← gen1;
   !LSH al,-6;
   RETURN(al) END.

(astruc) PROCEDURE(astrng); %a-string to upper case%
   % convert a-string (in astrng) to upper case %
   %-----------------%
   LOCAL length, bytptr, char;
   REF astrng;
   IF (length ← astrng.L) = empty THEN RETURN(&astrng);
   bytptr ← chbptr(empty) + &astrng;
   DO IF (char ← ↑bytptr) IN ['a,'z] THEN
      .bytptr ← char - 40B
   UNTIL (length ← length - 1) = empty;
   RETURN(&astrng);
   END.

(slngth) PROCEDURE (bp1, bp2); %string length%
   %returns the lenght of the string  represented by the passed byte
   pointers, the first of which is assumed to be set so that an
   increment and load byte will get first char of the string and the
   second pointing to the last char in the string.  NOTE: the index
   field is ignored and the only the size field of bp1 is used!%

   %bpadr, bpsize, bpbitpos%
   %-----------------%
   LOCAL length;
   length ← (bp2.bpadr - bp1.bpadr)*(36 / bp1.bpsize) +
      (bp1.bpbitpos - bp2.bpbitpos) / bp1.bpsize;
   RETURN(MAX(length, 0));
   END.

(srmake) %make string from value%
   PROCEDURE(value,astrng,base);
   LOCAL power, char;
   REF astrng;
   IF value < 0 THEN
      BEGIN
      IF &astrng THEN *astrng* ← *astrng*, '-
      ELSE apachr('-);
      value ← -value;
      END;
   power ← 1;
   WHILE power*base <= value DO power ← power*base;
   UNTIL power < 1 DO
      BEGIN
```

```
      DIV value / power, char, value;                                    0786
      IF &astrng THEN *astrng* ← *astrng*, char + '0                     0787
      ELSE apachr(char + '0);                                            0788
      power ← power / base;                                              0789
      END;                                                               0790
   RETURN END.                                                           0791

(srmk) PROCEDURE(value,base);                                            0792
   %In string construction use apachr to append string to sar.%          0793
   srmake(value, 0, base);                                               0794
   RETURN END.                                                           0795

(srval) PROCEDURE(astrng, base);                                         0796
   %convert string to value%                                             0797
   LOCAL value, cnt, char;                                               0798
   REF astrng;                                                           0799
   value ← 0;                                                            0800
   cnt ← 1;                                                              0801
   UNTIL (char ← *astrng*[cnt]) = ENDCHR DO                              0802
      BEGIN                                                              0803
      value ← value*base + char - '0;                                    0804
      BUMP cnt;                                                          0805
      END;                                                               0806
   RETURN (value) END.                                                   0807

DECLARE reltab = (0,0,0, %no 0 relational%                               0817
   TRUE, FALSE, FALSE, %1: <%                                            0818
   FALSE, TRUE, FALSE, %2: =%                                            0819
   TRUE, TRUE, FALSE, %3: <=%                                            0820
   0, 0, 0, %no 4 relational%                                            0821
   FALSE, TRUE, TRUE, %5: >=%                                            0822
   TRUE, FALSE, TRUE, %6: #%                                             0823
   FALSE, FALSE, TRUE); %7: >%                                           0824

(repstr) %The routine replaces the characters in one a-string,
OLDAST, with those in another, NEWAST, beginning at CHARNO. If the
new a-string is empty, the routine returns FALSE; otherwise it
returns TRUE.%                                                           0825
   PROCEDURE (newast, oldast, charno);                                   0826
   %--------------------%                                               0827
   LOCAL astr1, repcnt;                                                  0828
   REF newast, oldast;                                                   0829
   IF (astr1 ← newast.L) = empty THEN RETURN (FALSE);                    0830
   IF charno + newast.L > oldast.L THEN                                  0831
      BEGIN                                                              0832
      IF charno + newast.L > oldast.M THEN                               0833
         RETURN (FALSE);                                                 0834
      oldast.L ← charno + newast.L;                                      0835
      END;                                                               0836
   repcnt ← empty + 1;                                                   0837
   UNTIL repcnt > astr1 DO                                               0838
      BEGIN                                                              0839
      *oldast*[charno] ← *newast*[repcnt];                               0840
      BUMP charno, repcnt;                                               0841
      END;                                                               0842
```

```
    RETURN (TRUE)                                                       0843
    END.
                                                                        0844
(srlset) %To set an A-string to a single character, call this
procedure with the character and the address of the string.%
                                                                        0858
    PROCEDURE (char, ast);                                              0859
    /ast/.L ← empty;                                                    0860
    apchr (char, ast);                                                  0861
    RETURN END.
                                                                        0862
(mkbptr) %Make Byte Pointer with the specified position, size and
address.  The address is 23 bits wide to allow indexing and
indirection.%
                                                                        0863
    PROCEDURE (pos, size, addr);                                        0864
    a1 ← addr; !LSHC a1,-24;                                            0865
    a1 ← size; !LSHC a 1,-6;                                             0866
    a1 ← pos;  !LSHC a1,-6;                                             0867
    RETURN (a2) END.
                                                                        0868
% Statement scanning & cntent-analysis support%                        0924
(pdc) PROCEDURE (pdcnum, pntloc);                                       0931
    %pointer decrement. arguments are number of postions to             0932
    move and address of pointer to be decremented.  %
                                                                        0933
    LOCAL                                                               0934
        end;   %count at end of statement%                              0935
    IF scndir = forward THEN                                            0936
        /pntloc+1/ ← MAX(/pntloc+1/-pdcnum, 1)                          0937
    ELSE                                                                0938
        BEGIN                                                           0939
        cpfse(/pntloc/);                                                0940
        end ← a2;                                                       0941
        /pntloc+1/ ← MIN(/pntloc+1/+pdcnum, end);                       0942
        END;                                                            0943
    RETURN;                                                             0944
    END.
                                                                        0945
(dptr) PROCEDURE(pntloc);                                               0946
    pdc(1, pntloc);                                                     0947
    RETURN END.
                                                                        0948
(tstsr) PROCEDURE (ast);                                                0949
    %Test string.  Compare the T-string specified by SWORK with the
    string whose address is passed as arg.  Fetches characters from
    string by calling READC.  On a successful match does SKIP RETURN
    and SWORK is left pointing to the character after the pattern.
    If the match fails, normal return.%
                                                                        0950
    LOCAL                                                               0951
        cnt,    %counter for characters in test string%                 0952
        char,   %character from source string%                          0953
        tsp;    %pointer into test string%                              0954
    REF ast;                                                            0955
    cnt ← ast.L;                                                        0956
```

```
   tsp ← chbmty + &ast;                                                    0957
   UNTIL (cnt ← cnt-1) < 0 DO                                              0958
      IF (char ← READC) = ENDCHR OR ↑tsp # char THEN RETURN;               0959
   SKIP RETURN END.
                                                                           0960
(tstf) PROCEDURE (ast);                                                    0961
   %Like tst except looks for any occurence of the test string in
   the remaining portion of the T-string.  This is done by first
   scanning thru looking for the first character of the string, then
   if find it the rest of the string is?compared.  If get a mismatch
   SWORK is reset to character?after the start of the current
   partial match and the process is repeated.  Failure occurs only
   when the T-string is exhausted.  Returns with SKIP RETURN on
   success, normal return on failure.%
                                                                           0962
   LOCAL                                                                   0963
      cnt,     %counter for characters in test string%                     0964
      cpos,    %value of swork1 where got match%                           0965
      nchar,   %number of characters in test string%                       0966
      char,    %character from source string%                              0967
      char1,   %first character of test string%                            0968
      tsp,     %pointer into the test string%                              0969
      tsp1;    %pointer to first character of test string%                 0970
   REF ast;                                                                0971
   nchar ← ast.L;                                                          0972
   IF nchar < 1 THEN SKIP RETURN;                                          0973
   tsp1 ← chbmty + &ast;                                                   0974
   char1 ← ↑tsp1; %first character of string%                              0975
   LOOP %scan until get match for first character in string%               0976
      CASE READC OF                                                        0977
         = ENDCHR : EXIT; %have reached the end%                           0978
         = char1 : %have a match on the first character%                   0979
            BEGIN                                                          0980
            cpos ← swork1;                                                 0981
            cnt ← 1;                                                       0982
            tsp ← tsp1;                                                    0983
            LOOP                                                           0984
               BEGIN                                                       0985
               IF (cnt ← cnt+1) > nchar THEN SKIP RETURN;                  0986
               IF (char ← READC) = ENDCHR THEN EXIT 2;                     0987
               IF ↑tsp # char THEN EXIT;                                   0988
               END;                                                        0989
            %failure -- back to scanning for first character%              0990
            swork1 ← cpos;                                                 0991
            fechc1(scndir, $swork);                                        0992
            END;                                                           0993
         ENDCASE;                                                          0994
   RETURN;                                                                 0995
   END.
                                                                           0996
(bfs) %branch false and scan %
                                                                           0997
   % Resets the character position, then reads another character
   from the statement.  If this is not an ENDCHR the character
   number is put on the stack and control is transfered to the
   location specified in the address of the pop.%
```

```
                                                                    0998
                                                                    0999
   PROCEDURE (brloc);                                              01000
   swork ← [p-1];                                                  01001
   swork1 ← [p];                                                   01002
   fechc1(scndir, $swork);                                         01003
   IF READC # ENDCHR THEN                                          01004
      BEGIN                                                        01005
      [p] ← swork1;                                                01006
      [m] ← brloc; %change return location%                        01007
      RETURN;                                                      01008
      END;                                                         01009
   p ← p - 2000002B;
   RETURN END.                                                     01010

(fxswork) PROCEDURE;                                               01011
   fechc1(scndir, $swork);                                         01012
   RETURN END.                                                     01013

(srsup) PROCEDURE; %dummy procedure for string support code%       01014
   %these are all called by JSP a4,x and return by JRST (a4)%      01015
   (fcp):                                                          01016
      swork ← a1;                                                  01017
      swork1 ← a2;                                                 01018
      !JRST (a4);                                                  01019
   (pcp):                                                          01020
      !PUSH p,swork;                                               01021
      !PUSH p,swork1;                                              01022
      !JRST (a4);                                                  01023
   (sptr):                                                         01024
      !MOVE  a2, swork;                                            01025
      !MOVEM a2,0(a1);                                             01026
      !MOVE  a2, swork1;                                           01027
      !MOVEM a2,1(a1);                                             01028
      !JRST (a4);                                                  01029
   (begarb):                                                       01030
      !PUSH p,swork;                                               01031
      !PUSH p,swork1;                                              01032
      !HLRZ a2,a1; %lower bound%                                   01033
      !PUSH p,a2;                                                  01034
      !HRRZ a2,a1; %upper bound%                                   01035
      !PUSH p,a2;                                                  01036
      a2 ← 0; %#ount%                                              01037
      !PUSH p,a2;                                                  01038
      !JRST (a4);                                                  01039
   (fxsp1): %leave a1 and a2 unchanged%                            01040
      sptr1 ← a1;                                                  01041
      a3 ← 1;                                                      01042
      sptr1[a3] ← a2;                                              01043
      !JRST (a4);                                                  01044
   (fxsp2):                                                        01045
      sptr2 ← a1;                                                  01046
      a3 ← 1;                                                      01047
      sptr2[a3] ← a2;                                              01048.
      !JRST (a4);                                                  01049
   (lpr):                                                          01050
      a1 ← [a2];                                                   01051
```

```
      a2 ← [a2+1];                                                    01052
      !JRST (a4);                                                     01053
   END.                                                               01054

EXTERNAL fcp, pcp, sptr, begarb, fxsp1, fxsp2, ipr;                   01055

(incarb) PROCEDURE;                                                   01056
   %update position on stack where last succeeded%                    01057
   [p-4] ← swork;                                                     01058
   [p-3] ← swork1;                                                    01059
   %increment count and compare to upper bound%                       01060
   RETURN(([p] ← [p]+1) < [p-1]) END.                                 01061

(endarb) PROCEDURE;                                                   01062
   p ← p - 3000003B;                                                  01063
   scp();                                                             01064
   p ← p - 2000002B;                                                  01065
   RETURN([p+5] IN [[p+3],[p+4]]) END.                                01066

(scp) PROCEDURE;                                                      01067
   swork ← [p-1];                                                     01068
   swork1 ← [p];                                                      01069
   fechc1(scndir, $swork);                                            01070
   RETURN END.                                                        01071

(scn1ft) PROCEDURE;                                                   01072
   fechc1(scndir ← 0, $swork);                                        01073
   RETURN END.                                                        01074

(scnrht) PROCEDURE;                                                   01075
   fechc1(scndir ← 1, $swork);                                        01076
   RETURN END.                                                        01077

(chrct) PROCEDURE (char, chrcls); %Character class test%             01078

   CASE chrcls OF                                                     01079
      =1: %CH%                                                        01080
         IF char IN [0,177B] THEN GOTO cctyes;                        01081
      =4: %LD%                                                        01082
         IF char IN ['A,'Z] OR                                        01083
            char IN ['a,'z] OR                                        01084
            char IN ['0,'9] THEN GOTO cctyes;                         01085
      =11: %NLD%                                                      01086
         IF char # ENDCHR AND                                         01087
            char NOT IN ['A,'Z] AND                                   01088
            char NOT IN ['a,'z] AND                                   01089
            char NOT IN ['0,'9] THEN GOTO cctyes;                     01090
      =7: %L%                                                         01091
         IF char IN ['a,'z] OR                                        01092
            char IN ['A,'Z] THEN GOTO cctyes;                         01093
      =8: %D%                                                         01094
         IF char IN ['0,'9] THEN GOTO cctyes;                         01095
      =9: %PT%                                                        01096
         IF char IN [15B,174B] THEN GOTO  CCTYES;                      01097
      =10: %NP%                                                       01098
         IF char # ENDCHR AND                                         01099
```

```
            char NOT IN [41B,174B] THEN GOTO cctyes;                    01100
         =5: %UL%                                                       01101
            IF char IN ['A,'Z] THEN GOTO cctyes;                        01102
         =6: %LL%                                                       01103
            IF char IN ['a,'z] THEN GOTO cctyes;                        01104
         =2: %ULD%                                                      01105
            IF char IN ['A,'Z] OR                                       01106
               char IN ['0,'9] THEN GOTO cctyes;                        01107
         =3: %LLD%                                                      01108
            IF char IN ['a,'z] OR                                       01109
               char IN ['0,'9] THEN GOTO cctyes;                        01110
         ENDCASE;                                                       01111
      RETURN (char);                                                    01112
      (cctyes): SKIP RETURN (char)                                      01113
      END.                                                              01114

   (cct) PROCEDURE(chrcls);                                             01115
      RETURN(SKIP chrct(READC, chrcls)) END.                            01116

   (cctf) PROCEDURE(chrcls);                                            01117
      WHILE (a1 ← READC) # ENDCHR DO                                    01118
         IF SKIP chrct(a1, chrcls) THEN RETURN(TRUE);                   01119
      RETURN (FALSE) END.                                               01120

   (span) PROCEDURE(chrcls);                                            01121
      LOCAL sw1;                                                        01122
      DO sw1 ← swork1 WHILE SKIP chrct(READC, chrcls);                  01123
      swork1 ← sw1;                                                     01124
      fxswork();                                                        01125
      RETURN END.                                                       01126

   (tchf) PROCEDURE(char);                                              01127
      CASE READC OF                                                     01128
         =char: RETURN(TRUE);                                           01129
         =ENDCHR: RETURN(FALSE);                                        01130
         ENDCASE REPEAT CASE;                                           01131
      END.                                                              01132

   (cpfse) PROCEDURE (stid);                                            01137
      %statement end t-pointer%                                         01138
      %called by SPL compiler only%                                     01139

      IF stid.stastr THEN                                               01141
         BEGIN                                                          01142
         a2 ← [stid.stadr].L + 1;                                       01143
         a1 ← stid;                                                     01144
         RETURN;                                                        01145
         END;                                                           01146
      flcpfse( stid );                                                  01147
      RETURN END.                                                       01154

% storage manipulation.  stacks, rings, buffers%                        01155
   DECLARE EXTERNAL FIELD                                               01156
      systkp = [0(rp), 18:0], %pointer%                                 01157
      systkt = [0(rp), 13:2], %type (1=stack, 2=ring)%                  01158
      systks = [1(rp), 18:18], %size of element%                        01159
```

```
   systke = [1(rp), 18:0]; %end of store%                                01160
   % rcpsh, rpop, rstsk, and <IDLIBE>gbotids use the fact that the
   stack body starts in word 2. %
                                                                         01161
(rcpsh) %Record push.  Arguments are (1) address of record, and (2)
address of stack or ring buffer.  The record is pushed on the
store.%
                                                                         01162
   PROCEDURE (recadr, st);                                               01163
   REF st;                                                               01164
   IF st.systkt NOT IN [1,2] THEN err(0);                                01165
   IF (st.systkp ← st.systkp + st.systks) = st.systke THEN               01166
      IF st.systkt = 1 THEN err(0$"Stack overflow") %stack overflow%
                                                                         01167
      ELSE st.systkp ← &st + 2; %ring wrap around%                       01168
   IF st.systks = 1 THEN [st.systkp] ← [recadr]                          01169
   ELSE mvbfbf (recadr, st.systkp, st.systks);                           01170
   RETURN END.
                                                                         01171
(rpop) %Record pop.  Argument is the address of a store.  Pop the
top element.%
                                                                         01172
   PROCEDURE (st);                                                       01173
   REF st;                                                               01174
   IF st.systkt = 1 %stack% THEN                                         01175
      IF st.systkp < &st+2 THEN err(0) %stack underflow%                 01176
      ELSE NULL                                                          01177
   ELSE IF st.systkt = 2 %ring% THEN                                     01178
      IF st.systkp <= &st+2 THEN                                         01179
         st.systkp ← st.systke                                           01180
      ELSE NULL                                                          01181
   ELSE err(0); %wrong type%                                             01182
   st.systkp ← st.systkp - st.systks;                                    01183
   RETURN END.
                                                                         01184
(rstsk) %Reset store%
                                                                         01185
   PROCEDURE (st);                                                       01186
   REF st;                                                               01187
   IF st.systkt NOT IN [1,2] THEN err(0);                                01188
   st.systkp ← &st + 2 - st.systks;                                      01189
   RETURN END.
                                                                         01190
(rcpto) %Record Pop To.  Arguments are (1) address of store, and (2)
address of record.  Pops the top record on the store to the
designated location.%
                                                                         01191
   PROCEDURE (st, recadr);                                               01192
   REF st;                                                               01193
   IF st.systkt NOT IN [1,2] THEN err(0);                                01194
   IF st.systks = 1 THEN [recadr] ← [st.systkp]                          01195
   ELSE mvbfbf (st.systkp, recadr, st.systks);                           01196
   rpop (&st);                                                           01197
   RETURN END.
                                                                         01198
FINISH                                                                   052
```