$DATA

```
< NLS, SDATA.NLS.7, >, 18-Apr-78 13:43 JDH ;;;;
FILE sdata   %  L10   <REL-NLS>SDATA   %% (L10,) (rel-nls,SDATA.rel,) %
                                                                      02
REGISTER %here so DDT will use these on printout%                     03
   r1=1, r2=2, r3=3, r4=4, r5=5, p=7, wa=8, s=9, m=10, rp=11,         04
   a1=12, a2=13, a3=14, a4=15;                                        05
%identification system%                                               06
   DECLARE EXTERNAL STRING                                            07
      userstr[50]; %string corresponding to login dir name% %PASSED%  08
% DECLARE's %                                                         09
   DECLARE EXTERNAL                                                  010
      entvec[8], %entry vector for nls%                              011
      tenex,  %tenex version number%                                 012
      lhostn,  %logical host number on network%                      013
      nojournalflag=0, %disable journal, ident checking%             067
      tops20flag=0,  %0 for tenex, 1 for tops20%                     083
      t2v3flag=0,  %1 for tops20 version 3%                          087
      nonetworkflag=0, %if nonzero, is "hostnumber" for non-network
      system%                                                        086
      gmtflag=0,  %for tops20, =1: running GMT as local time, ISID
      glitch%                                                        084
      lcltimzon=8,  %constant for Pacific time, but could be U.O.%   085
   %...illegal instruction pseudo-interrupt support...%              014
      ilsdsp,  %dispatch address%                                    015
      %DO NOT REORDER THE NEXT TWO DECLARATIONS%                     016
      psireg[15],                                                    017
      psirglast;  %registers at time of psi%                         068
   DECLARE EXTERNAL TEXT POINTER                                     018
      p1,                                                            019
      p2;                                                            069
   DECLARE EXTERNAL                                                  020
   %...storage management...%                                        026
      cgetblk,                                                       027
      cfreeblk,                                                      074
      cmakezone,                                                     073
      creplenish,                                                    072
      cunlink,                                                       071
      clink,                                                         070
   %...pseudointerrupt tables...%       %PASSED%                  028
      levtab=lev1lc,                                                 029
      levt2=lev2lc,                                                  076
      levt3=lev3lc,                                                  075
      lev1lc,                                                        030
      lev2lc,                                                        031
      lev3lc,                                                        032
      chntab[36],                                                    033
      savchntab[4], %for use in no-oping first four pseudo interrupts%
                                                                     034
      trpcnt= 0,                                                     035
      ccignore,  %count of how many times to ignore control characters%
                                                                     036
   %...pseudointerrupt data...% %PASSED%                             037
      sav40,                                                         052
      svac1,                                                         038
      svacs[15],                                                     078
      svacse,                                                        077
```

```
   %...system timer globals...%                                                  039
      timrset=0,    %timer is running flag%                                      040
      timrproc,     %procedure to be called when the timmer goes off%            041
      timra1,    % argument 1 for timrproc %                                     042
      timra2,    % argument 2 for timrproc %                                     043
      timra3,    % argument 3 for timrproc %                                     044
      timra4,    % argu1ent 4 for timrproc %                                     045
%...global display/print parameters...%                                         046
   %...display support...%                                                      047
      msglck,                                                                   048
      msgtim,                                                                   049
      msgfrk;                                                                   050
% for use with syntax generating commands %                                     053
   % stack and stack pointers for partial path generation %                     065
      DECLARE EXTERNAL                                                          066
         csstkx,                                                                079
         csstkb,                                                                081
         csstk[100];                                                            080
   % variables controlling output mode %                                       054
      DECLARE EXTERNAL                                                          055
         omode= 0,      % output modes %                                        056
            % 0 - debugging: do typeas in onode %                              057
            % 1 - inserting: do insert statement in onode %                     058
               % cinstid = stid of statement to insert after %                  059
               % ilevel = level to insert at %                                 060
            % 2 - being used as seqgen: do send in onode %                     061
               % instid = adr of seq work area %                               062
            % 3 - being used in show type command: call fbctl in onode %      063
         cinstid,                                                               064
         ilevel;                                                                082

FINISH of sdata                                                                 051
```

SELECT

```
< NLS, SELECT.NLS;14, >, 19-FEB-77 17:17 KJM ;;;;
FILE select % L10 <rel-nls>select %% (110,) (rel-nls,select.rel,) %    02
   % DECLARATIONS %                                                    03
      REF tda, inpt, fbstr, sysmsg;                                    04
   % SELECTION PROCESSORS %                                           047
      % SELECTION ROUTINES %                                          048
         ( xselect) PROCEDURE(                                       03446
            % process a selection %                                   03447
            % FORMAL ARGUMENTS %                                      03448
               resultptr,        % ptr to the return argument record %
                                                                      03449
               parsemode,        % parsing mode %                     03450
               %test%                                                 03451
               type,     % address of type code for selection %      03452
               curptr );         % ptr to path stack entry %         03453
            % NORMAL RETURNS %                                        03454
               % 2 text pointers are returned in the result record which
               delemit the selection %                                03455
            % ABNORMAL RETURNS %                                      03456
               % a signal is generated whenever a CD is encounterd %
                                                                      03457
            LOCAL % VARIABLES %                                      03458
               bugaction,                % what to do when a BUG is typed
               %                                                      03459
               bugs,             % number of bug selections required %
                                                                      03460
               caflag,           % TRUE if CA reguried after help %  03461
               char,             % inpt char %                       03462
               defaction,                % what to do on typing any other
               char %                                                 03463
               delimiter,                % entity delimiter routine %
                                                                      03464
               function,                 % processing function %     03465
               i,                        % number of selections obtained
               %                                                      03466
               ltda,             % ptr to current display area %     03467
               oaction,          % what to do when a OPT char is typed %
                                                                      03468
               optionstr,                % str containing current options
               %                                                      03469
               promptsav,                % ptr to prompts saving string %
                                                                      03470
               statesav,                 % ptr to state saving area %
                                                                      03471
               temp,             % scratch variable %                03472
               tptr;             % ptr to a result textptr %         03473
            LOCAL STRING                                             03474
               helpstr[200];     % help feedback string %            03475
            REF % VARIABLES %                                        03476
               bugaction,                                            03477
               curptr,                                               03478
               defaction,                                            03479
               delimiter,                                            03480
               function,                                             03481
               ltda,                                                 03482
               oaction,                                              03483
```

```
    promptsav,                                                    03484
    resultptr,                                                    03485
    statesav,                                                     03486
    tptr,                                                         03487
    type;                                                         03488
REF tda, inpt, fbstr, sysmsg;                                     03489
%-----------------%                                              03490
% trap error signals in order to recover from selection
errors without aborting the command %                             03491
   ON SIGNAL                                                      03492
      = errsig:                                                   03493
         BEGIN                                                    03494
         % output the error message %                             03495
            dismes( 2, MESSAGE);                                  03496
            &sysmsg _ $"";                                        03497
         % invoke the xselect routine in cleanup mode to
         reset any bugs, etc that it may have put up %            03498
            xselect( &resultptr, cleanup);                        03499
         % reset the feedback %                                   03500
            cflstr.L _ statesav.cfl1len;                          03501
            fbctl( fbpop );                                       03502
         % reset selection count %                                03503
            statesav.nselects _ 0;                                03504
         % resume gathering of a selection %                      03505
            GOTO selectagain;                                     03506
         END;                                                     03507
   ELSE;                                                          03508
(selectagain):                                                    03509
% set ptr to selection state record %                             03510
   &statesav _ &resultptr + psellen;                              03511
% clear the needconfirm flag %                                   03512
   needconfirm _ FALSE;                                           03513
% clear special control character $echo flag %                   03514
   ctrlchar _ FALSE;                                              03515
% process according to parsemode %                               03516
   CASE parsemode OF                                              03517
      = parsing:    % normal parsing mode %                       03518
         BEGIN                                                    03519
         slink _ cdlnk _ clpsw _ nwlnk _ FALSE;                   03520
         CASE type OF                                             03521
            % STRUCTURAL ENTITIES %                               03522
               = 1 %- branch -%:                                  03523
                  BEGIN                                           03524
                  bugs _ 1;                                       03525
                  &delimiter _ $stdr;                             03526
                  END;                                            03527
               = 2 %- group -%:                                   03528
                  BEGIN                                           03529
                  bugs _ 2;                                       03530
                  &delimiter _ $grpdr;                            03531
                  END;                                            03532
               = 3 %- plex -%:                                    03533
                  BEGIN                                           03534
                  bugs _ 1;                                       03535
                  &delimiter _ $plxdr;                            03536
                  END;                                            03537
```

```
        = 4 %- statement -%:                                    03538
            BEGIN                                               03539
            bugs _ 1;                                           03540
            &delimiter _ $stdr;                                 03541
            END;                                                03542
        % TEXTUAL ENTITIES %                                    03543
        = 5 %- character -%:                                    03544
            BEGIN                                               03545
            &delimiter _ $cdr;                                  03546
            bugs _ 1;                                           03547
            END;                                                03548
        = 6 %- controlchar -%:                                  03549
            BEGIN                                               03550
            &delimiter _ $tdr;                                  03551
            bugs _ 2;                                           03552
            ctrlchar _ TRUE; % special echos %                  03553
            END;                                                03554
        = 7 %- invisible -%:                                    03555
            BEGIN                                               03556
            &delimiter _ $idr;                                  03557
            bugs _ 1;                                           03558
            END;                                                03559
        = 8 %- link -%:                                         03560
            BEGIN                                               03561
            slink _ TRUE;                                       03562
            cdlnk _ TRUE;                                       03563
            &delimiter _ $ldr;                                  03564
            bugs _ 1;                                           03565
            END;                                                03566
        = 9 %- directory -%:                                    03567
            BEGIN                                               03568
            slink _ TRUE;                                       03569
            &delimiter _ $vdr;                                  03570
            bugs _ 1;                                           03571
            END;                                                03572
        = 10 %- password -%:                                    03573
            BEGIN                                               03574
            clpsw _ TRUE;                                       03575
            &delimiter _ $vdr;                                  03576
            bugs _ 1;                                           03577
            END;                                                03578
        = 11 %- number -%:                                      03579
            BEGIN                                               03580
            &delimiter _ $ndr;                                  03581
            bugs _ 1;                                           03582
            END;                                                03583
        = 12 %- text -%:                                        03584
            BEGIN                                               03585
            &delimiter _ $tdr;                                  03586
            bugs _ 2;                                           03587
            END;                                                03588
        = 13 %- visible -%:                                     03589
            BEGIN                                               03590
            &delimiter _ $vdr;                                  03591
            bugs _ 1;                                           03592
            END;                                                03593
```

```
            = 14 %- word -%:                                    03594
                BEGIN                                           03595
                &delimiter _ $wdr;                              03596
                bugs _ 1;                                       03597
                END;                                            03598
            % MISC. ENTITIES %                                  03599
            = 15 %- file -%:                                    03600
                BEGIN                                           03601
                slink _ cdlnk _ TRUE;                           03602
                &delimiter _ $flndr;                            03603
                bugs _ 1;                                       03604
                END;                                            03605
            = 16 %- newfilelink -%:                             03606
                BEGIN                                           03607
                slink _ cdlnk _ nwlnk _ TRUE;                   03608
                &delimiter _ $ldr;                              03609
                bugs _ 1;                                       03610
                END;                                            03611
            = 17 %- oldfilelink -%:                             03612
                BEGIN                                           03613
                slink _ cdlnk _ TRUE;                           03614
                &delimiter _ $ldr;                              03615
                bugs _ 1;                                       03616
                END;                                            03617
            = 18 %- name -%:                                    03618
                BEGIN                                           03619
                &delimiter _ $nmdr;                             03620
                bugs _ 1;                                       03621
                END;                                            03622
            = 19 %- ident -%:                                   03623
                BEGIN                                           03624
                &delimiter _ $iddr;                             03625
                bugs _ 1;                                       03626
                END;                                            03627
            = 20 %- identlist -%:                               03628
                BEGIN                                           03629
                &delimiter _ $idldr;                            03630
                bugs _ 2;                                       03631
                END;                                            03632
            = 21 %- edge -%:                                    03633
                BEGIN                                           03634
                &delimiter _ 0;                                 03635
                bugs _ 1;                                       03636
                END;                                            03637
            = 22 %- marker -%:                                  03638
                BEGIN                                           03639
                &delimiter _ $wdr;                              03640
                bugs _ 1;                                       03641
                END;                                            03642
        ENDCASE SIGNAL (interperr, $"Unknown Entity
        selection type");                                       03643
        % we now set up actions to be accompilshed whenever
        a trigger character is encountered .  The
        characters currently recognized as trigger
        characters are :                                        03644
           1) CA character-- may be a bug selection.            03645
```

```
2) $option char-- indicated what to do when an
$option is typed.                                   03646
3) "? character:  request for some prompting.
                                                    03647
4) any other char: --this is the default action
for the type of selection. %                        03648
% the action function is set to 0 if it is not
permitted %                                         03649
caflag _ FALSE;                                     03650
CASE [curptr.curnodeptr].opcode OF                  03651
   = $dsel:                                         03652
      BEGIN                                         03653
      &defaction _ $getdae;                         03654
      &oaction _ 0;                                 03655
      IF nlmode = fulldisplay THEN                  03656
         BEGIN                                      03657
         IF type = 21 %+ edge +% THEN               03658
            BEGIN                                   03659
            &defaction _ 0;                         03660
            &bugaction _ $getwindow;                03661
            optionstr _ $"BUG";                     03662
            END                                     03663
         ELSE                                       03664
            BEGIN                                   03665
            &bugaction _ $getbug;                   03666
            optionstr _  $"BUG   or   ADDRESS";
                                                    03667
            END;                                    03668
         caflag _ TRUE;                             03669
         arm();                                     03670
         END                                        03671
      ELSE                                          03672
         BEGIN                                      03673
         &bugaction _ $getdae;                      03674
         optionstr _ $"ADDRESS";                    03675
         END;                                       03676
      END;                                          03677
   = $ssel:                                         03678
      BEGIN                                         03679
      &defaction _ $getdae;                         03680
      &oaction _ (CASE type OF                      03681
         = 19 %- ident -%: $getid;                  03682
         = 20 %- identlist -%: $getidlist;          03683
         ENDCASE $getlit);                          03684
      IF nlmode = fulldisplay THEN                  03685
         BEGIN                                      03686
         &bugaction _ $getbug;                      03687
         optionstr _ $"BUG   or   ADDRESS   or
         OPTION TYPEIN";                            03688
         caflag _ TRUE;                             03689
         arm();                                     03690
         END                                        03691
      ELSE                                          03692
         BEGIN                                      03693
         &bugaction _ $getdae;                      03694
         optionstr _ $"ADDRESS  or  OPTION
```

```
                    TYPEIN";                                    03695
                    END;                                        03696
                 END;                                           03697
              = $1sel:                                          03698
                 BEGIN                                          03699
                 &defaction _ (CASE type OF                     03700
                    = 19 %- ident -%: $getid;                   03701
                    = 20 %- identlist -%: $getidlist;           03702
                    ENDCASE $getlit);                           03703
                 &oaction _ $getdae;                            03704
                 IF nlmode = fulldisplay THEN                   03705
                    BEGIN                                       03706
                    &bugaction _ $getbug;                       03707
                    optionstr _ $"TYPEIN   or   BUG   or
                    OPTION ADDRESS";                            03708
                    caflag _ TRUE;                              03709
                    arm();                                      03710
                    END                                         03711
                 ELSE                                           03712
                    BEGIN                                       03713
                    &bugaction _ $getlit;                       03714
                    optionstr _ $"TYPEIN   or   OPTION
                    ADDRESS";                                   03715
                    END;                                        03716
                 END;                                           03717
              ENDCASE SIGNAL (interperr, $"Unknown
              selection type");                                 03718
           % initialize the selection count i %                 03719
            i _ statesav.nselects+1;                            03720
           % save away entity type in the state record %        03721
            statesav.entype _ type;                             03722
            statesav.maxselect _ bugs;                          03723
           % get the reqired number of selections %             03724
            LOOP                                                03725
               BEGIN                                            03726
               % set tptr to point to the result text
               pointer for the selection and save away the
               prompt string %                                  03727
                  IF i = 1                                      03728
                     THEN                                       03729
                        BEGIN                                   03730
                        statesav.cfl1len _ cflstr.L;            03731
                        &tptr _  &resultptr;                    03732
                        &promptsav _ &statesav +
                        $selstatesize;                          03733
                        promptsav.M _ 30; promptsav.L _ 0;
                                                                03734
                        IF prompstr.L > 30 THEN prompstr.L _
                        30;                                     03735
                        *promptsav* _ *prompstr*;               03736
                        END                                     03737
                     ELSE                                       03738
                        BEGIN                                   03739
                        statesav.cfl2len _ cflstr.L;            03740
                        &tptr _ &resultptr + 2;                 03741
                        END;                                    03742
```

```
% prompt the user if appropriate:  The prompt
string was put together by evaluate after
calling sleuth too look ahead in the grammar.
It was aslo output there if a branching
decision had to be made at that point. %
                                                03743
   IF inprompts # noprompts AND NOT cueflg
                                                03744
      THEN fbctl( incues, $prompstr );          03745
% set the processing function according to
what is typed next %                            03746
   CASE char _ lookc() OF                       03747
      = cachar, = inschar, = rptchar:           03748
         &function _ &bugaction;                03749
      = optchar:% address selection %           03750
         BEGIN                                  03751
         inpt();                                03752
         IF ([curptr.curnodeptr].opcode =
         $dsel) AND (type = 21 %+ edge +%)
         THEN                                   03753
            BEGIN                               03754
            fbctl( '? );                        03755
            REPEAT CASE;                        03756
            END                                 03757
         ELSE &function _ &oaction;             03758
         END;                                   03759
      = '?:  % request for help %               03760
         BEGIN                                  03761
         IF msjfn THEN  %monitoring commands%
                                                03762
            msrecord(mquestion, 0, 0);          03763
         inpt();                                03764
         cueflg _ FALSE;                        03765
         *helpstr* _ "Please specify a ",
         *[[&type + 1]]*, " by
             ", *[optionstr]*, EOL, "or
         <CTRL-Q> for HELP, or <CTRL-S> for
         SYNTAX", EOL;                          03766
         fbctl( IF caflag THEN typecalit ELSE
         typelit, $helpstr );                   03767
         REPEAT CASE;                           03768
         END;                                   03769
      = 'S-100B:  % <^S> request for SYNTAX %
                                                03770
         BEGIN                                  03771
         inpt();                                03772
         cueflg _ FALSE;                        03773
         cshelp(
         $pathstk+pathx-$totalrecsize,
         cmdmode, FALSE);                       03774
         REPEAT CASE;                           03775
         END;                                   03776
      = CD:                                     03777
         BEGIN                                  03778
         inpt();                                03779
         SIGNAL (cmdelete);                     03780
```

```
                END;                                    03781
            = BC,                                       03782
            = BW:                                       03783
                BEGIN                                   03784
                inpt();                                 03785
                SIGNAL (popstate);                      03786
                END;                                    03787
            ENDCASE                                     03788
                &function _ &defaction;                 03789
        % save away the function address in the state
        record %                                        03790
           IF i = 1                                     03791
              THEN statesav.sel1fun _ &function         03792
              ELSE statesav.sel2fun _ &function;
                                                        03793
        % $call the processing function or put out a
        diagnostic %                                    03794
           CASE &function OF                            03795
              = $getbug,                                03796
              = $getwindow:                             03797
                BEGIN                                   03798
                function( parsemode, &tptr );           03799
                IF msjfn THEN  %monitoring commands%
                                                        03800
                   msrecord(mbugmark, 0, 0);            03801
                END;                                    03802
              = $getdae:                                03803
                BEGIN                                   03804
                IF i = 1 THEN                           03805
                   BEGIN                                03806
                   % initialize text ptr for first
                   selection to the current marker
                   for the current display area %
                                                        03807
                   &ltda _ lda();                       03808
                   IF auxinput THEN                     04320
                      BEGIN %processing commands --
                      force like TNLS addressing%
                                                        04321
                      tptr _ curmkr;                    04322
                      tptr[1] _ curmkr[1];              04323
                      END                               04324
                   ELSE                                 04325
                      BEGIN %normal input -- get
                      TNLS- or DNLS-type addressing%
                                                        04326
                      tptr _ ltda.dacsp;                04327
                      tptr[1] _ ltda.daccnt;            04328
                      END;                              04329
                   END                                  03811
                ELSE                                    03812
                   BEGIN                                03813
                   % initialize the text pointer for
                   the second address to be the
                   result obtained in the first
                   selection %                          03814
```

```
                        tptr _ [&tptr - 2];                 03815
                        tptr[1] _ [&tptr - 1];              03816
                        END;                                03817
                     function( parsemode, &tptr );          03818
                     END;                                   03819
                  = $getlit, = $getid, = $getidlist:
                                                            03820
                     BEGIN                                  03821
                     IF i # 1 THEN REPEAT CASE (0);         03822
                     % save a count of the actual number
                     of selections obtained %               03823
                        statesav.nselects _ i;              03824
                     function( parsemode, &resultptr,
                     &resultptr+2 );                        03825
                     IF msjfn THEN  %monitoring commands%
                                                            03826
                        msrecord(mliteral, 0, %length%
                        resultptr[3]-resultptr[1]);         03827
                     EXIT LOOP;                             03828
                     END;                                   03829
                  = 0:   % error %                          03830
                     BEGIN                                  03831
                     inpt();                                03832
                     fbctl( '? );                           03833
                     REPEAT LOOP;                           03834
                     END;                                   03835
               ENDCASE;                                     03836
            % save a count of the actual number of
            selections obtained %                           03837
               statesav.nselects _ i;                       03838
            % set up for another selection if required %
                                                            03839
               IF i >= bugs THEN EXIT LOOP;                 03840
               BUMP i;                                      03841
               cueflg _ FALSE;                              03842
               fbctl( echostr, $"through" );                03843
               arm();                                       03844
            END;                                            03845
         % invoke the delimiter routine to delimit the
         selection %                                        03846
            CASE &function OF                               03847
               = $getlit, = $getid, = $getidlist:           03848
                  BEGIN                                     03849
                  % set tptr to point to second delimiter in
                  string %                                  03850
                     &tptr _ &resultptr + d2sel;            03851
                  % edit delimiters onto typed in file names
                  or links if not already given %           03852
                     CASE type OF                           03853
                        = 17 %- oldfilelink -%, = 16 %-
                        newfilelink -%:                     03854
                           littolnk( TRUE, &resultptr,
                           &tptr);                          03855
                        = 8 %- link -%:                     03856
                           littolnk( FALSE, &resultptr,
                           &tptr);                          03857
```

```
                        ENDCASE;                                    03858
                    END;                                            03859
                ENDCASE                                             03860
                    IF &delimiter # 0 THEN                          03861
                        BEGIN                                       03862
                        CASE i OF                                   03863
                            = 1: % single selection %               03864
                               delimiter(&resultptr, &resultptr,
                               &resultptr+2 );                      03865
                            = 2: % double selection %               03866
                            delimiter(&resultptr, &resultptr+2,
                            &resultptr, &resultptr+2);              03867
                            ENDCASE;                                03868
                        END;                                        03869
    END;                                                            03870
= backup,       % FALSE parse backup %                            03871
= cleanup:      % TRUE parsing termination %                      03872
    BEGIN                                                           03873
    FOR i _ statesav.nselects DOWN UNTIL < 1 DO                     03874
        BEGIN                                                       03875
        &function _                                                 03876
            IF i = 1                                                03877
                THEN statesav.sel1fun                               03878
                ELSE statesav.sel2fun;                              03879
        function(parsemode, &resultptr+2*(i-1) );                   03880
        END;                                                        03881
    ctrlchar _ FALSE;                                               03882
    END;                                                            03883
= popselect:  % pop selection %                                     03884
    BEGIN                                                           03885
    slink _ cdlnk _ clpsw _ FALSE;                                  03886
    % set up the full actual argument list %                       03887
        &curptr _ &resultptr - $pathrecsize;                        03888
        temp _ statesav.entype;                                     03889
    % delete the last selection %                                  03890
        i _ statesav.nselects;                                      03891
        IF i = 1                                                    03892
            THEN                                                    03893
                BEGIN                                               03894
                cflstr.L _ statesav.cfl1len;                        03895
                &function _ statesav.sel1fun;                       03896
                END                                                 03897
            ELSE                                                    03898
                BEGIN                                               03899
                cflstr.L _ statesav.cfl2len;                        03900
                &function _ statesav.sel2fun;                       03901
                END;                                                03902
        function(cleanup, &resultptr+2*(i-1) );                     03903
    % clean up the feedback %                                      03904
        &promptsav _ &statesav + $selstatesize;                     03905
        *promptstr* _ *promptsav*;                                  03906
        fbctl( fbpop );                                             03907
    % decrement the selection count by 1 %                         03908
        statesav.nselects _ i-1;                                    03909
    % invoke the selection processor recursively to
    gather another selection %                                      03910
```

```
                xselect( &resultptr, parsing, $temp, &curptr);
                                                                    03911
            END;                                                    03912
         ENDCASE;                                                   03913
         RETURN (&resultptr);                                       03914
      END.
                                                                    03915
   (getlit) PROCEDURE( % get a typed literal string %               02142
      % FORMAL ARGUMENTS %                                          02143
         parsemode,        % parsing mode %                        02144
         tptr1,    % address of starting tptr %                   02145
         tptr2);  % address of finishing tptr %                    02146
      % NORMAL RETURNS %                                           02147
         % none %                                                  02148
      % ABNORMAL RETURNS %                                        02528
         % a "popstate" SIGNAL is generated if BC or BW is typed
         and the literal string is empty %                          02529
         % a "statesig" SIGNAL is generated if getstring is out of
         storage space %                                            02530
      LOCAL % VARIABLES %                                         02151
         selstateptr,      % ptr to selstate record %             02152
         strptr;  % ptr to literal collection string %            02153
      REF % VARIABLES %                                           02154
         strptr,                                                  02155
         selstateptr,                                             02156
         tptr1,                                                   02157
         tptr2;                                                   02158
      % --------------- %                                         02159
      % set up ptr to the selection state record %                02160
         &selstateptr _ &tptr1 + psellen;                         02161
      CASE parsemode OF                                           02162
         = parsing:                                               02163
            BEGIN                                                 02164
            % allocate a literal collection string %             02531
               selstateptr.litptr _ &strptr _ 0;                  02532
               ON SIGNAL ELSE                                     02533
                  BEGIN                                           02534
                  dismes(1,$"Fatal Storage Error: Please Reset");
                                                                    02535
                  clrbuf(FALSE);  % clear input buffers%          02536
                  SIGNAL(statesig);                               02537
                  END;                                            02538
               &strptr _ getstring( 2000, $dspblk );              02539
               ON SIGNAL ELSE;                                    02540
            % save ptr to allocated string in the selstate record
            %                                                     02168
               selstateptr.litptr _ &strptr;                      02169
            litreset _ FALSE;                                     02170
            % do prompting for text %                            02180
               fbctl( incues, $"T:");                             02181
            % look for a null literal %                          02171
               IF lookc() = $ctln THEN                            02172
                  BEGIN                                           02173
                  inpt();                                         02174
                  FIND SF(*strptr*) ^tptr1 ^tptr2;                02175
                  selstateptr.dolitreset _ FALSE;                 02176
```

```
                needconfirm _ TRUE;                                    02177
                RETURN;                                                02178
                END;                                                   02179
             % read the literal %                                       02182
                IF nlmode = fulldisplay                                02183
                   THEN selstateptr.dolitreset _ TRUE                  02184
                   ELSE                                                02185
                      BEGIN                                            02186
                      echolt();                                        02187
                      selstateptr.dolitreset _ FALSE;                  02188
                      END;                                             02189
                ON SIGNAL ELSE litapflag _ TRUE;                       02552
                setlit();                                              02190
                CASE rdlit( &strptr, 0) OF                             02191
                   = 3: % BC or BW typed and string already empty %    02192
                      SIGNAL (popstate);                               02193
                   ENDCASE;                                            02194
                ON SIGNAL ELSE;                                        02553
                IF nlmode = fulldisplay                                02195
                   THEN                                                02196
                      BEGIN                                            02197
                      IF littakedown THEN                              02198
                         BEGIN                                         02199
                         rstlit();                                     02200
                         selstateptr.dolitreset _ FALSE;               02201
                         END;                                          02202
                      END                                              02203
                   ELSE echoff();                                      02204
             % set the text ptr to point to the string %               02205
                FIND SF(*strptr*) ^tptr1 SE(*strptr*) ^tptr2;          02206
             END;                                                      02207
          = backup,                                                    02208
          = cleanup:                                                   02209
             BEGIN                                                     02210
             &selstateptr _ &tptr1 + 4;                                02211
             &strptr _ selstateptr.litptr;                             02212
             IF &strptr THEN freestring( &strptr, $dspblk);            02213
             IF selstateptr.dolitreset                                 02214
                AND littakedown                                        02215
                THEN rstlit(); % reset literal feedback area %         02216
             END;                                                      02217
       ENDCASE;                                                        02218
       RETURN;                                                         02219
       END.                                                            02220

    (getid) PROCEDURE( % get a typed ident string %                    02763
       % FORMAL ARGUMENTS %                                            02764
          parsemode,       % parsing mode %                           02765
          tptr1,   % address of starting tptr %                       02766
          tptr2);  % address of finishing tptr %                      02767
       % NORMAL RETURNS %                                             02768
          % none %                                                     02769
       % ABNORMAL RETURNS %                                           02770
          % a "popstate" SIGNAL is generated if BC or BW is typed
```

```
   and the literal string is empty %                                    02771
   % a "statesig" SIGNAL is generated if getstring is out of
   storage space %                                                      02772
LOCAL % VARIABLES %                                                     02773
   selstateptr,     % ptr to selstate record %                         02774
   strptr;  % ptr to literal collection string %                       02775
REF % VARIABLES %                                                       02776
   strptr,                                                              02777
   selstateptr,                                                         02778
   tptr1,                                                               02779
   tptr2;                                                               02780
% --------------- %                                                    02781
% set up ptr to the selection state record %                           02782
   &selstateptr _ &tptr1 + psellen;                                     02783
CASE parsemode OF                                                       02784
   = parsing:                                                           02785
      BEGIN                                                             02786
      % allocate a literal collection string %                          02787
         selstateptr.litptr _ &strptr _ 0;                              02788
         ON SIGNAL ELSE                                                 02789
            BEGIN                                                       02790
            dismes(1,$"Fatal Storage Error: Please Reset");
                                                                        02791
            clrbuf(FALSE);  % clear input buffers%                      02792
            SIGNAL(statesig);                                           02793
            END;                                                        02794
         &strptr _ getstring( 2000, $dspblk );                          02795
         ON SIGNAL ELSE;                                                02796
      % save ptr to allocated string in the selstate record
      %                                                                 02797
         selstateptr.litptr _ &strptr;                                  02798
      LOOP                                                              02945
         BEGIN                                                          02946
         litreset _ FALSE;                                              02799
         % do prompting for text %                                      02800
            fbctl( incues, $"T:");                                      02801
         % look for a null literal %                                    02802
            IF lookc() = $ctln THEN                                     02803
               BEGIN                                                    02804
               inpt();                                                  02805
               (nullid):                                                03276
               FIND SF(*strptr*) ^tptr1 ^tptr2;                         02806
               selstateptr.dolitreset _ FALSE;                          02807
               needconfirm _ TRUE;                                      02808
               RETURN;                                                  02809
               END;                                                     02810
            % read the literal %                                        02811
            IF nlmode = fulldisplay                                     02812
               THEN selstateptr.dolitreset _ TRUE                       02813
               ELSE                                                     02814
                  BEGIN                                                 02815
                  echolt();                                             02816
                  selstateptr.dolitreset _ FALSE;                       02817
                  END;                                                  02818
            ON SIGNAL ELSE litapflag _ TRUE;                            02819
            setlit();                                                   02820
```

```
                IF NOT idfno THEN                                   03401
                   BEGIN  %open it and note that we did%            03402
                   idfno _ loaidfil();                              03403
                   setidfno _ TRUE;                                 03404
                   END;                                             03405
                CASE getlid(&strptr) OF                             03265
                   = 1: %got a good ident%                          03266
                   = 2: %query failed, try again%                   03268
                      BEGIN                                         03271
                      *strptr* _ NULL;                              03273
                      fbctl(fbaddlit, $"
                      Please type another IDENT: ");                03352
                      echoff();                                     04316
                      REPEAT LOOP;                                  03274
                      END;                                          03272
                   = 3: %back up the command%                       03269
                      SIGNAL(popstate);                             03275
                   = 4: %null ident%                                03270
                      BEGIN                                         03279
                      rstlit();                                     03278
                      *strptr* _ NULL;                              03281
                      GOTO nullid;                                  03277
                      END;                                          03280
                   ENDCASE err($"Undefined return from getlid
                   detected in getid");                             03267
                IF setidfno AND idfno THEN                          03406
                   BEGIN  %close it since we opened it%             03407
                   setidfno _ FALSE;                                03408
                   [flntadr(idfno)].flnoclose _ FALSE;              03409
                   close(idfno := 0);                               03410
                   END;                                             03411
                IF nlmode = fulldisplay                             02826
                   THEN                                             02827
                      BEGIN                                         02828
                      IF littakedown THEN                           02829
                         BEGIN                                      02830
                         rstlit();                                  02831
                         selstateptr.dolitreset _ FALSE;            02832
                         END;                                       02833
                      END                                           02834
                   ELSE echoff();                                   02835
             % set the text ptr to point to the string %            02836
               FIND SF(*strptr*) ^tptr1 SE(*strptr*) ^tptr2;
                                                                    02837
             EXIT LOOP;                                             02948
             END;                                                   02947
          END;                                                      02838
       = backup,                                                    02839
       = cleanup:                                                   02840
          BEGIN                                                     02841
          &selstateptr _ &tptr1 + 4;                                02842
          &strptr _ selstateptr.litptr;                             02843
          IF &strptr THEN freestring( &strptr, $dspblk);            02844
          IF selstateptr.dolitreset                                 02845
             AND littakedown                                        02846
             THEN rstlit(); %reset literal feedback area%           02847
```

```
            IF setidfno AND idfno THEN                                  03422
               BEGIN  %close it since we opened it%                     03423
               setidfno _ FALSE;                                        03424
               [flntadr(idfno)].flnoclose _ FALSE;                      03425
               close(idfno := 0);                                       03426
               END;                                                     03427
            END;                                                        02848
         ENDCASE;                                                       02849
         RETURN;                                                        02850
         END.
                                                                        02851
      (getlid) PROCEDURE( % get a typed ident string and do ident
      query if requested -- subordinate to GETID and GETIDLIST %
                                                                        03158
         % FORMAL ARGUMENTS %                                           03159
            idstr);  %string for collecting ident%                      03160
         % RETURNS %                                                    03163
            % 1, with idstr having an ident in it %                     03164
            % 2, with idstr null -- query failed %                      03259
            % 3, BC or BW with string empty %                           03258
            % 4, no ident collected -- end of ident list %              03263
         LOCAL i;                                                       03331
         REF % VARIABLES %                                              03171
            idstr;                                                      03175
         % --------------- %                                           03176
         % read the literal %                                          03208
            CASE rdlit( &idstr, $idntdel) OF                            03218
               = 3: % BC or BW typed and string already empty %         03219
                  RETURN(3); %back space old string%                    03220
               ENDCASE;                                                 03221
            LOOP                                                        03351
               IF idstr.L THEN                                          03223
                  IF identquery(&idstr) THEN RETURN(1) %successful,
                  ident in idstr%                                       03224
                  ELSE %user query was unsucessful%                     03262
                     BEGIN                                              03225
                     IF idstr.L THEN REPEAT LOOP;                       03226
                     RETURN(2); %query failed -- no new ident
                     specified%                                         03228
                     END                                                03229
               ELSE RETURN(4); %end of ident list or null ident
               collected%                                               03261
         END.
                                                                        03257
      (getidlist) PROCEDURE( % get a typed ident list string %          02852
         % FORMAL ARGUMENTS %                                           03024
            parsemode,       % parsing mode %                         03025
            tptr1,   % address of starting tptr %                       03026
            tptr2);  % address of finishing tptr %                     03027
         % NORMAL RETURNS %                                            03028
            % none %                                                   03029
         % ABNORMAL RETURNS %                                          03030
            % a "popstate" SIGNAL is generated if BC or BW is typed
            and the literal string is empty %                           03031
            % a "statesig" SIGNAL is generated if getstring is out of
            storage space %                                             03032
```

```
LOCAL % VARIABLES %                                                    03033
   selstateptr,     % ptr to selstate record %                          03034
   strptr;  % ptr to literal collection string %                       03035
LOCAL TEXT POINTER z1, z2;                                             03332
LOCAL STRING                                                           03138
   work[50];  %for ident queries%                                      03139
REF % VARIABLES %                                                      03036
   strptr,                                                             03037
   selstateptr,                                                        03038
   tptr1,                                                              03039
   tptr2;                                                              03040
% --------------- %                                                   03041
% set up ptr to the selection state record %                          03042
   &selstateptr _ &tptr1 + psellen;                                    03043
CASE parsemode OF                                                      03044
   = parsing:                                                          03045
      BEGIN                                                            03046
      % allocate a literal collection string %                         03047
         selstateptr.litptr _ &strptr _ 0;                             03048
         ON SIGNAL ELSE                                                03049
            BEGIN                                                      03050
            dismes(1,$"Fatal Storage Error: Please Reset");
                                                                       03051
            clrbuf(FALSE);  % clear input buffers%                     03052
            SIGNAL(statesig);                                          03053
            END;                                                       03054
         &strptr _ getstring( 2000, $dspblk );                         03055
         ON SIGNAL ELSE;                                               03056
      % save ptr to allocated string in the selstate record
      %                                                                03057
         selstateptr.litptr _ &strptr;                                 03058
      litreset _ FALSE;                                                03061
      % do prompting for text %                                        03062
         fbctl( incues, $"T:");                                        03063
      % look for a null literal %                                      03064
         IF lookc() = $ctln THEN                                       03065
            BEGIN                                                      03066
            inpt();                                                    03067
            FIND SF(*strptr*) ^tptr1 ^tptr2;                           03068
            selstateptr.dolitreset _ FALSE;                            03069
            needconfirm _ TRUE;                                        03070
            RETURN;                                                    03071
            END;                                                       03072
      % read the literal %                                             03073
         IF nlmode = fulldisplay                                       03074
            THEN selstateptr.dolitreset _ TRUE                         03075
            ELSE                                                       03076
               BEGIN                                                   03077
               echolt();                                               03078
               selstateptr.dolitreset _ FALSE;                         03079
               END;                                                    03080
         setlit();                                                     03082
         DO                                                            03119
            BEGIN                                                      03120
            ON SIGNAL ELSE litapflag _ TRUE;                           03081
            CASE rdlit( &strptr, 0) OF                                 03083
```

```
            = 3: % BC or BW typed and string already
            empty %                                          03084
               SIGNAL (popstate);                            03085
            ENDCASE;                                         03086
         ON SIGNAL ELSE;                                     03087
         FIND SF(*strptr*) ^z2;                              03307
         IF NOT idfno THEN                                   03428
            BEGIN  %open it and note that we did%            03429
            idfno _ loaidfil();                              03430
            setidfno _ TRUE;                                 03431
            END;                                             03432
         WHILE (FIND z2 ['"/".] < CH > ^z1) DO               03301
            BEGIN                                            03302
            IF FIND '" THEN FIND ['"/ENDCHR] ^z2             03304
            ELSE FIND CH $(LD / '' / SP / '-) $'. ^z2;
                                                             03308
            *work* _ z1 z2;                                  03306
            IF work.L THEN                                   03088
               BEGIN                                         03311
               %                                             03335
               IF nlmode = fulldisplay THEN rstlit()
                                                             03315
               ELSE crlf();                                  03320
               %                                             03336
               IF identquery($work) THEN %user query was
               sucessful%                                    03089
                  ST z1 z2 _ *work*                          03090
               ELSE                                          03310
                  BEGIN                                      03328
                  FIND z2 > $(SP/',) ^z2;                    03334
                  ST z1 z2 _ NULL;                           03327
                  END;                                       03329
               FIND z1 ^z2;                                  03333
               END;                                          03312
            END;                                             03303
         IF setidfno AND idfno THEN                          03433
            BEGIN  %close it since we opened it%             03434
            setidfno _ FALSE;                                03435
            [flntadr(idfno)].flnoclose _ FALSE;              03436
            close(idfno := 0);                               03437
            END;                                             03438
         IF *strptr*[strptr.L] NOT= SP THEN                  03131
            *strptr* _ *strptr*, SP;                         03133
         END                                                 03121
      UNTIL curchr = CA;                                     03122
      WHILE *strptr*[strptr.L] = SP DO                       03134
         BUMP DOWN strptr.L;                                 03135
      IF nlmode = fulldisplay                                03091
         THEN                                                03092
            BEGIN                                            03093
            IF littakedown THEN                              03094
               BEGIN                                         03095
               rstlit();                                     03096
               selstateptr.dolitreset _ FALSE;               03097
               END;                                          03098
            END                                              03099
```

```
                ELSE echoff();                                          03100
            % set the text ptr to point to the string %                 03101
              FIND SF(*strptr*) ^tptr1 SE(*strptr*) ^tptr2;             03102
            END;                                                        03105
         = backup,                                                      03106
         = cleanup:                                                     03107
            BEGIN                                                       03108
            &selstateptr _ &tptr1 + 4;                                  03109
            &strptr _ selstateptr.litptr;                               03110
            IF &strptr THEN freestring( &strptr, $dspblk);              03111
            IF selstateptr.dolitreset                                   03112
               AND littakedown                                          03113
               THEN rstlit(); %reset literal feedback area%             03114
            IF setidfno AND idfno THEN                                  03439
               BEGIN  %close it since we opened it%                     03440
               setidfno _ FALSE;                                        03441
               [flntadr(idfno)].flnoclose _ FALSE;                      03442
               close(idfno := 0);                                       03443
               END;                                                     03444
            END;                                                        03115
         ENDCASE;                                                       03116
      RETURN;                                                           03117
      END.
                                                                        03118
   (identquery) PROCEDURE(idstr); %process Ident Queries%               02950
      LOCAL type;                                                       02951
      LOCAL TEXT POINTER tptr1, tptr2;                                  02952
      LOCAL STRING work[500];                                           02953
      REF idstr;                                                        02954
      IF idstr.L = empty THEN RETURN(TRUE);                             02955
      CASE *idstr*[1] OF                                                02956
         =".: %lastname search%                                         02957
            BEGIN                                                       02958
            *idstr* _ *idstr*[2 TO idstr.L];                            02964
            CCPOS SE(*idstr*);  %check for incomplete name%             02965
            IF READC = ". THEN                                          02966
               BEGIN %partially specified last name, remove
               trailing periods%                                        02967
               DO BUMP DOWN idstr.L UNTIL READC NOT= ".;                02969
               type _ idchr;                                            02970
               END                                                      02971
            ELSE type _ lname;                                          02972
            stnamcap(&idstr);                                           02973
            RETURN( idflsearch(type, &idstr, idfno) );                  02974
            END;                                                        02975
         ='": %search name fields of individuals, groups and orgs%
                                                                        02976
            BEGIN                                                       02977
            FIND SF(*idstr*) CH ^tptr1 ( ['"] ^tptr2 _tptr2 /
            SE(*idstr*) ^tptr2);                                        02983
            *idstr* _ tptr1 tptr2;                                      02984
            RETURN( idflsearch(strinname, &idstr, idfno) );             02985
            END;                                                        02986
         = SP: %deblank front of string%                                02994
            BEGIN                                                       02995
            *idstr* _ *idstr*[2 TO idstr.L];                            02996
```

```
            REPEAT CASE;                                              02997
            END;                                                      02998
         ENDCASE;                                                     02999
      RETURN(TRUE);                                                   03022
      END.
                                                                      03023
   (getbug) PROCEDURE( % get a typed literal string %                  0516
      % FORMAL ARGUMENTS %                                            0517
         parsemode,        % parsing mode %                           0518
         tptr);    % address of result tptr %                        0519
      % NORMAL RETURNS %                                              0520
         % none %                                                     0521
      % ABNORMAL RETURNS %                                            0522
         % a "popstate" SIGNAL is generated if BC or BW is typed
         and the literal string is empty %                            0523
      REF % VARIABLES %                                              0524
         tptr;                                                        0525
      % --------------- %                                            0526
      CASE parsemode OF                                              0527
         = parsing:                                                  0528
            BEGIN                                                    0529
            needconfirm _ TRUE;                                      0530
            INPUT BUG tptr;                                          0531
            disarm();                                                0532
            END;                                                     0533
         = backup,                                                   0534
         = cleanup:                                                  0535
            BEGIN                                                    0536
            delbm(); % delete last bug mark %                        0537
            disarm();                                                0538
            END;                                                     0539
      ENDCASE;                                                       0540
      RETURN;                                                        0541
      END.
                                                                     0542
   (getdae) PROCEDURE( % get a typed literal string %                 0543
      % FORMAL ARGUMENTS %                                           0544
         parsemode,        % parsing mode %                          0545
         tptr);    % address of result tptr %                       0546
      % NORMAL RETURNS %                                             0547
         % none %                                                    0548
      % ABNORMAL RETURNS %                                           0549
         % a "popstate" SIGNAL is generated if BC or BW is typed
         and the literal string is empty %                           0550
      REF % VARIABLES %                                             0551
         tptr;                                                       0552
      % --------------- %                                           0553
      CASE parsemode OF                                             0554
         = parsing:                                                 0555
            getaddr(&tptr);                                         0556
         = backup,                                                  0557
         = cleanup:                                                 0558
            IF nlmode = fulldisplay                                 0559
               THEN dn( $""); % clear name buffer %                 0560
      ENDCASE;                                                      0561
      RETURN;                                                       0562
```

```
    END.
                                                                    0563
(getwindow) PROCEDURE( % get a window selection %                   04330
  % FORMAL ARGUMENTS %                                              04331
    parsemode,       % parsing mode %                              04332
    tptr);   % address of result tptr %                            04333
  % NORMAL RETURNS %                                                04334
    % none %                                                        04335
  LOCAL % variables %                                               04336
    da,      % ptr to display rea %                                04337
    cords1,  % display window coords %                            04338
    x,       % x coordinate %                                      04339
    y,       % y coordinate %                                      04340
    char,    % character being bugged %                            04341
    r;       % division remainder %                                04342
  LOCAL TEXT POINTER tp;                                            04343
  REF % VARIABLES %                                                 04344
    tptr, da, inpt;                                                 04345
  % --------------- %                                              04346
  CASE parsemode OF                                                 04347
    = parsing:                                                      04348
      BEGIN                                                         04349
      % check for proper mode and read over the ca character
      %                                                             04350
        IF nlmode = typewriter                                      04351
          OR inpt() # cachar THEN                                   04352
          err( $"Illegal use of getwindow routine");                04353
      % get the display area address, x and y screen
      coordinates %                                                 04354
        IF NOT &da _ dsparea(lcda(:cords1)) THEN                    04355
          err($"lcda failed, getwindow");                           04356
        % X coordinate %                                           04357
          x _ cords1.xcord - da.daleft;                             04358
          DIV x / da.dahinc, x, r;                                  04359
          IF r AND r >= (da.dahinc/2) THEN BUMP x;                  04360
          x _ x * da.dahinc;                                        04361
          IF x >= da.daright - da.daleft THEN                       04362
            x _ da.daright - da.daleft - da.dahinc;                 04363
              %otherwise MARKIT will fail%                          04364
          cords1.xcord _ x + da.daleft;                             04365
        % y coordinate %                                           04366
          y _ cords1.ycord - da.datop;                              04367
          DIV y / da.davinc, y, r;                                  04368
          IF r AND r >= (da.davinc/2) THEN BUMP y;                  04369
          y _ y * da.davinc;                                        04370
          IF y >= da.dabottom - da.datop THEN                       04371
            y _ da.dabottom - da.datop - da.davinc;                 04372
              %otherwise MARKIT will fail%                          04373
          cords1.ycord _ y + da.datop;                              04374
      %put bug mark on screen%                                      04375
        ON SIGNAL ELSE                                              04393
          BEGIN                                                     04394
          char _ SP;                                                04395
          GOTO markbug;                                             04396
          END;                                                      04397
        tp _ fndchr(&da, x, y: tp[1]);                              04376
```

```
                ON SIGNAL ELSE;                                    04398
                CCPOS tp;                                          04377
                char _ READC;                                      04378
                (markbug):  %for goto from signal%                 04399
                markit(&da, x, y, char);                           04379
             % stuff display area address and translated screen
             coordinates into the result tptr %                    04380
                tptr _ &da;                                        04381
                tptr[1] _ cords1;                                  04382
             % set the needconfirm flag so we won't back up over
             the confirmation %                                    04383
                needconfirm _ TRUE;                                04384
             END;                                                  04385
          = backup,                                                04386
          = cleanup:                                               04387
             % delete the bug mark from the screen %               04388
             delbm();                                              04389
       ENDCASE;                                                    04390
       RETURN;                                                     04391
       END.
                                                                   04392
% DELIMITER ROUTINES %                                             0614
   (stdr) PROCEDURE( % truncates a bug selection to a statement
   selection %                                                     0615
      bug,        % text ptr for bug %                           0616
      tptr1,% first text pointer %                               0617
      tptr2);% second text pointer %                             0618
      REF bug, tptr1, tptr2;                                     0619
      %--------------%                                           0620
      FIND SF(bug) ^tptr1 SE(bug) ^tptr2;                        0621
      RETURN;                                                    0622
      END.
                                                                 0623
   (grpdr) PROCEDURE( % delimits a structural group %            0624
      % FORMAL ARGUMENTS %                                       0625
         tptr1,   % addrss of first text pointer %               0626
         tptr2,   % address of second text pointer %             0627
         ret1,    % address of the first result text pointer %
                                                                 0628
         ret2);   % address of the second result text pointer %
                                                                 0629
      % RETURNS %                                               0630
         % text pointers ret1 and ret2 are set up to delimit the
         group identified by the bug selections tptr1 and tptr2 %
                                                                 0631
      REF % VARIABLES %                                          0632
         tptr1, tptr2, ret1, ret2;                               0633
      %------------%                                            0634
      ret1[1] _ ret2[1] _ 1;                                     0635
      ret1 _ grptst( tptr1, tptr2: ret2);                        0636
      RETURN;                                                    0637
      END.
                                                                 0638
   (plxdr) PROCEDURE( % delimits a structural plex %             0639
      % FORMAL ARGUMENTS %                                       0640
         tptr1,   % addrss of first text pointer %               0641
```

```
      ret1,    % address of the first result text pointer %
                                                                0642
      ret2);   % address of the second result text pointer %
                                                                0643
    % RETURNS %                                                 0644
      % text pointers ret1 and ret2 are set up to delimit the
      plex identified by the bug selection tptr1 %              0645
    REF % VARIABLES %                                           0646
      tptr1, ret1, ret2;                                        0647
    %------------%                                             0648
    ret1[1] _ ret2[1] _ 1;                                      0649
    ret1 _ plxset( tptr1: ret2);                                0650
    RETURN;                                                     0651
    END.
                                                                0652
% LEVELADJ STRING %                                           0653
  (xlevadj) PROCEDURE( % collect leveladj string %            03916
    % FORMAL ARGUMENTS %                                      03917
      resultptr,       % ptr to the return argument record %
                                                               03918
      parsemode);% parsing mode %                              03919
    % NORMAL RETURNS %                                        03920
      % 1) returns the leveladj count (d = -1, u = +1 etc ) %
                                                               03921
      % 2) returns the leveladj string %                      03922
    % ABNORMAL RETURNS %                                      03923
    LOCAL % VARIABLES %                                       03924
      char,    % inpt char %                                  03925
      levadjstr;       % collection string for leveladj %    03926
    REF % VARIABLES %                                         03927
      levadjstr,                                              03928
      resultptr;                                              03929
    REF tda, inpt, fbstr, sysmsg;                             03930
    %-----------------%                                       03931
    % set up a string descriptor to point to the level adjust
    collection string %                                       03932
      &levadjstr _ &resultptr + plvlen;                       03933
    % process according to parsemode %                       03934
      CASE parsemode OF                                       03935
        = parsing:                                            03936
          BEGIN                                               03937
          % initialize the count to 0 %                       03938
            resultptr _ 0;                                    03939
          % initialize string lengths %                       03940
            levadjstr.M _ 25;                                 03941
            levadjstr.L _ 0;                                  03942
          % if no level adjust then just exit %              03943
            IF nolevadj THEN RETURN(&resultptr);              03944
          % prompt the user for a level adj %                03945
            fbctl( incues, $"L:" );                           03946
          % collect the level adjust string %                03947
            LOOP                                              03948
              BEGIN                                           03949
              IF nlmode = fulldisplay                         03950
                THEN dn( &levadjstr );                        03951
              CASE char _ inpt() OF                           03952
```

```
=SP, =cachar, =inschar, =rptchar: %
terminators %                                   03953
   BEGIN                                        03954
   IF msjfn THEN  %monitoring commands%
                                                03955
      msrecord(mlevadj, 0, levadjstr.L);
                                                03956
   EXIT LOOP;                                   03957
   END;                                         03958
='u:                                            03959
   BEGIN                                        03960
   BUMP resultptr;                              03961
   *levadjstr* _ *levadjstr*, char;             03962
   IF nlmode # fulldisplay                      03963
      THEN typech( char );                      03964
   END;                                         03965
='d:                                            03966
   BEGIN                                        03967
   BUMP DOWN resultptr;                         03968
   *levadjstr* _ *levadjstr*, char;             03969
   IF nlmode # fulldisplay                      03970
      THEN typech( char );                      03971
   END;                                         03972
='?:                                            03973
   BEGIN                                        03974
   IF msjfn THEN  %monitoring commands%
                                                03975
      msrecord(mquestion, 0, 0);                03976
   fbctl( typelit, $"Please type a
   LEVEL-ADJUST string:
         u / d / <SPACE> / <CA>
   or <CTRL-Q> for HELP, or <CTRL-S> for
   SYNTAX");                                    03977
   END;                                         03978
= 'S-100B:  % <^S> request for SYNATX %
                                                03979
   cshelp( $pathstk+pathx-$totalrecsize,
   cmdmode, FALSE);                             03980
=BC:                                            03981
   IF levadjstr.L > 0                           03982
      THEN                                      03983
         BEGIN                                  03984
         CASE *levadjstr*[levadjstr.L] OF
                                                03985
            = 'd: BUMP resultptr;               03986
            = 'u: BUMP DOWN resultptr;
                                                03987
            ENDCASE;                            03988
         IF nlmode # fulldisplay THEN
                                                03989
            BEGIN                               03990
            typech(BC);                         03991
            typech( *levadjstr*[
            levadjstr.L ] );                    03992
            END;                                03993
         levadjstr.L _ levadjstr.L - 1;
```

```
                                                                    03994
                                    END                             03995
                                ELSE                                03996
                                    SIGNAL(popstate);               03997
                            =BW:                                    03998
                                IF levadjstr.L > 0                  03999
                                THEN                                04000
                                    BEGIN                           04001
                                    IF nlmode # fulldisplay         04002
                                        THEN typech(BW);            04003
                                    *levadjstr* _ NULL;             04004
                                    resultptr _ 0;                  04005
                                    END                             04006
                                ELSE                                04007
                                    SIGNAL(popstate);               04008
                            =CD:                                    04009
                                SIGNAL (cmdelete);                  04010
                        ENDCASE                                     04011
                            BEGIN                                   04012
                            % return the processed chars to the inpt
                            buffer %                                04013
                                unput();                            04014
                                IF levadjstr.L > 0 THEN resetb(
                                &levadjstr );                       04015
                            % clear result string %                 04016
                                *levadjstr* _ NULL;                 04017
                            % reset the count %                     04018
                                resultptr _ 0;                      04019
                            % clear name area IF DNLS %             04020
                                IF nlmode = fulldisplay             04021
                                    THEN dn( &levadjstr );          04022
                            EXIT LOOP;                              04023
                            END;                                    04024
                        END;                                        04025
                    IF nlmode = typewriter THEN                     04026
                        BEGIN                                       04027
                        crlf();                                     04028
                        END;                                        04029
                    END;                                            04030
                = backup,                                           04031
                = cleanup:                                          04032
                    BEGIN                                           04033
                    resultptr _ 0;                                  04034
                    IF nlmode = fulldisplay THEN                    04035
                        BEGIN                                       04036
                        *levadjstr* _ NULL;                         04037
                        dn( &levadjstr );                           04038
                        END;                                        04039
                    END;                                            04040
            ENDCASE;                                                04041
        RETURN (&resultptr);                                        04042
        END.

                                                                    04043
% VIEWSPEC STRING %                                                 0761
    (xviewspecs) PROCEDURE( % processes viewspecs %                 04044
        % the current viewspecs are modified by the current inpt
```

```
string.  when a delimiter for the viewspec string is
encountered, the updated viewspecs are returned.  Note that
the new viewspecs are not saved in the display area viewspec
words %                                                    04045
% FORMAL ARGUMENTS %                                      04046
   resultptr,       % ptr to the return argument record %
                                                           04047
   parsemode);% parsing mode %                             04048
% NORMAL RETURNS %                                        04049
   % 1) pvsrecord %                                        04050
   % 2)... viewspec collection string %                    04051
% ABNORMAL RETURNS %                                      04052
LOCAL % VARIABLES %                                        04053
   da,      % ptr to display area %                        04054
   vspec[2],        % viewspec words %                     04055
   save,    % flag save word %                             04056
   sav1,    % viewspec save word for recreate display %
                                                           04057
   sav2,    % viewspec save word for recreate display %
                                                           04058
   vstr,    % collection string for viewspecs %            04059
   goodvs,  % flag set for good viewspecs %                04060
   char;    % current inpt char %                          04061
REF % VARIABLES %                                          04062
   da,                                                     04063
   vstr,                                                   04064
   resultptr;                                              04065
REF tda, inpt, fbstr, sysmsg;                              04066
%-----------------%                                        04067
% set vstr to point to the viewspec collection string %
                                                           04068
   &vstr _ &resultptr + pvslen;                            04069
% process according to parsemode %                        04070
   CASE parsemode OF                                       04071
      = parsing:                                           04072
         BEGIN                                             04073
         % initialize the viewspec collection string %    04074
            vstr.L _ 0; vstr.M _ 20;                       04075
         % set da to point to the current display area %
                                                           04076
            &da _ lda();                                   04077
         % save user's display area progs in the vsrecord %
                                                           04078
            resultptr.vscacode _ cspcacode;                04079
            resultptr.vsusqcod _ cspusqcod;                04080
         % if no viewspecs then just return current ones %
                                                           04081
            IF novspec THEN                                04082
               BEGIN                                       04083
               resultptr _ da.davspec;                     04084
               resultptr[1] _ da.davspc2;                  04085
               RETURN(&resultptr);                         04086
               END;                                        04087
         % save the current viewspecs in the viewspec save
         area %                                            04088
            vspec _ IF cspupdate THEN cspvs ELSE da.davspec;
```

```
                                                                    04089
    vspec[1] _ IF cspupdate THEN cspvs[1] ELSE
    da.davspc2;                                                     04090
% prompt the user for viewspecs %                                   04091
    fbctl( incues, $"V:");                                          04092
% set up for viewspec collection %                                  04093
    RESET savevspec;                                                04094
    IF nlmode = fulldisplay THEN                                    04095
       BEGIN                                                        04096
       dspvsp( vspec, vspec[1], 3); % display them %
                                                                    04097
       ltl();                                                       04098
       af();                                                        04099
       END                                                          04100
    ELSE                                                            04101
       BEGIN                                                        04102
       IF buffs # buffn THEN typebuff();                            04103
       echolt();                                                    04104
       END;                                                         04105
% collect the viewspec string %                                     04106
LOOP                                                                04107
   BEGIN                                                            04108
   CASE char _ inpt() OF                                            04109
      =cachar, =inschar, =rptchar: % terminators %
                                                                    04110
         BEGIN                                                      04111
         % set return viewspec words %                             04112
            resultptr _ vspec;                                      04113
            resultptr[1] _ vspec[1];                                04114
         % clean up the feedback %                                 04115
            IF nlmode = fulldisplay THEN                            04116
               BEGIN                                                04117
               lts();                                               04118
               an ();                                               04119
               END                                                  04120
            ELSE echoff();                                          04121
         IF msjfn THEN  %monitoring commands%                       04122
            msrecord(mviewspecs, 0, vstr.L);                        04123
         EXIT LOOP;                                                 04124
         END;                                                       04125
      =CD:                                                          04126
         BEGIN                                                      04127
         % clean up the feedback %                                 04128
            IF nlmode = fulldisplay THEN                            04129
               BEGIN                                                04130
               dspvsp( da.davspec, da.davspc2, 3);
                                                                    04131
               lts();                                               04132
               an ();                                               04133
               END                                                  04134
            ELSE echoff();                                          04135
         SIGNAL (cmdelete);                                         04136
         END;                                                       04137
      ='f:                                                          04138
         IF nlmode = fulldisplay                                    04139
            THEN                                                    04140
```

```
                        BEGIN                                   04141
                        IF vspec.vsrlev THEN                    04142
                           vspec _ <SEQGEN, reslev>(vspec,
                           getlev(da.dacsp));                   04143
                        sav1 _ da.davspec := vspec;             04144
                        sav2 _ da.davspc2 := vspec[1];          04145
                        da.davspec.vsdaft := TRUE;              04146
                        dafrmt (&da, 0);                        04147
                        da.davspec _ sav1;                      04148
                        da.davspc2 _ sav2;                      04149
                        IF NOT sav1.vsdaft THEN
                        dspvsp(da.davspec, da.davspc2, 3);
                                                                04150
                        END;                                    04151
                  =BC:                                          04152
                     IF vstr.L > 0                              04153
                        THEN                                    04154
                           BEGIN                                04155
                           IF nlmode # fulldisplay THEN         04156
                              BEGIN % $echo deleted chars %
                                                                04157
                              typech( BC );                     04158
                              typech( *vstr* [vstr.L] );        04159
                              END;                              04160
                           bkc( &vstr );                        04161
                           POP savevspec TO vspec;              04162
                           END                                  04163
                        ELSE                                    04164
                           BEGIN                                04165
                           SIGNAL (popstate);                   04166
                           END;                                 04167
                     ='?:                                       04168
                        BEGIN                                   04169
                        IF msjfn THEN  %monitoring commands%    04170
                           msrecord(mquestion, 0, 0);           04171
                        fbctl( typelit, $"Please type in VIEWSPECS
                        or <CTRL-Q> for HELP or <CTRL-S> for
                        SYNTAX");                               04172
                        END;                                    04173
                     = 'S-100B:  % <^S> request for SYNATX %    04174
                        cshelp( $pathstk+pathx-$totalrecsize,
                        cmdmode, FALSE);                        04175
               ENDCASE                                          04176
                  BEGIN                                         04177
                  % the current viewpsec words (vspec) are
                  pushed onto the savevspec stack and setlt is
                  called to put the current viewspec into
                  effect %                                      04178
                     PUSH vspec ON savevspec;                   04179
                     vspec _ setlt (char, vspec, vspec[1] :
                     vspec[1], goodvs);                         04180
                     IF NOT goodvs THEN dismes(2, $"Undefined
                     viewspec -- ignored");                     04181
                  % append the current character to the
                  viewspec string %                             04182
                     *vstr* _ *vstr*, char;                     04183
```

```
                    END;                                          04184
                % the viewspec display is updated if in DNLS %
                                                                  04185
                    IF nlmode = fulldisplay THEN                  04186
                        BEGIN                                     04187
                        dn($vstr);                                04188
                        dspvsp (vspec, vspec[1], 3);              04189
                        END;                                      04190
                END; % of LOOP %                                  04191
                                                                  04192
                END;                                              04193
            = backup,                                             04194
            = cleanup:                                            04195
                BEGIN                                             04196
                % if no viewspecs then just exit %                04197
                    IF novspec THEN EXIT CASE;                    04198
                % clean up the feedback %                         04199
                    IF nlmode = fulldisplay THEN                  04200
                        BEGIN                                     04201
                        % cleanup the display %                   04202
                            %Bad code:  da record not updated until
                            cmdfinish, so old values get displayed by
                            this branch (but new values still in
                            viewspec record)                      04203
                            &da _ lda();                          04204
                            dspvsp( da.davspec, da.davspc2, 3);   04205
                            %                                     04206
                        lts();                                    04207
                        an ();                                    04208
                        dn ($"");                                 04209
                        END                                       04210
                    ELSE echoff();                                04211
                    END;                                          04212
        ENDCASE;                                                  04213
    RETURN (&resultptr);                                          04214
    END.
                                                                  04215
% COMMAND CONFIRMATION RECOGNIZER %                                0908
  (xconfirm) PROCEDURE( % process confirmation character %        04216
    % FORMAL ARGUMENTS %                                          04217
        resultptr,       % ptr to the return argument record %
                                                                  04218
        parsemode,       % parsing mode %                        04219
        lastopcode);% opcode of previous interpreter inst. %
                                                                  04220
    % NORMAL RETURNS %                                           04221
        % 1) returns the command completion code %                04222
        % the current command completion code is saved in the
        global complcode and has the following values %           04223
            % = 1: normal command completion %                    04224
            % = 2: insert statement mode %                        04225
            % = 3: repeat command mode %                          04226
    % ABNORMAL RETURNS %                                         04227
    REF % VARIABLES %                                             04228
        resultptr;                                                04229
        REF tda, inpt, fbstr, sysmsg;                             04230
```

```
        %-----------------%                                          04231
        % process according to parsemode %                         04232
           CASE parsemode OF                                        04233
              = parsing:                                            04234
                 BEGIN                                              04235
                 % check to see in the last inst was a $call on a
                 built in recognition function.  If it was, then we
                 back up the inpt buffer by 1 character and use the
                 terminator character of that recognizer as the inpt
                 character for the confirmation character recognizer
                 %                                                  04236
                    IF lastopcode IN [$ssel, $levadj]               04237
                       AND buffs = buffn % inpt buffer is empty %
                                                                    04238
                       AND NOT needconfirm                          04239
                       THEN unput()                                 04240
                       ELSE fbctl(incues, $"OK:" );                 04241
                 % process the next character, looking for a
                 terminator %                                       04242
                    CASE inpt() OF                                  04243
                       = rptchar:                                   04244
                          complcode _ 3;                            04245
                       = inschar:                                   04246
                          complcode _ 2;                            04247
                       = cachar:                                    04248
                          NULL;                                     04249
                       = BC, = BW:                                  04250
                          SIGNAL (popstate);                        04251
                       = CD:                                        04252
                          SIGNAL (cmdelete);                        04253
                       = '?:                                        04254
                          BEGIN                                     04255
                          IF msjfn THEN  %monitoring commands%      04256
                             msrecord(mquestion, 0, 0);             04257
                          fbctl( typelit, $"Please type an OK
                          character or <CTRL-Q> for HELP or <CTRL-S>
                          for SYNTAX");                             04258
                          REPEAT CASE;                              04259
                          END;                                      04260
                       = 'S-100B:  % <^S> request for SYNATX %      04261
                          BEGIN                                     04262
                          cshelp( $pathstk+pathx-$totalrecsize,
                          cmdmode, FALSE);                          04263
                          REPEAT CASE;                              04264
                          END;                                      04265
                    ENDCASE                                         04266
                       BEGIN                                        04267
                       fbctl( '? );                                 04268
                       REPEAT CASE;                                 04269
                       END;                                         04270
                 resultptr _ complcode;                             04271
                 END;                                               04272
           ENDCASE;                                                 04273
        RETURN (&resultptr);                                        04274
        END.
                                                                    04275
```

```
% entity finding routines - known as delimiters %                    0961
   (cdr) PROCEDURE (bug,ptr1,ptr2);                                  0962
      REF bug, ptr1, ptr2;                                           0963
      FIND bug > ^ptr1 CH ^ptr2;                                     0964
      RETURN;                                                        0965
      END.                                                           0966
                                                                     0967
   (idr) PROCEDURE (bug,ptr1,ptr2);                                  02554
      LOCAL TEXT POINTER tp1;                                        02555
      REF bug, ptr1, ptr2;                                           02556
      FIND bug ^tp1 < $NP ^ptr1 tp1 > CH $NP ^ptr2;                  02557
      RETURN;                                                        02558
      END.                                                           02559
                                                                     02560
   (tdr) PROCEDURE (bug1,bug2,ptr1,ptr2);                            01998
      REF bug1, bug2, ptr1, ptr2;                                    01999
      LOCAL TEXT POINTER tp1;                                        02000
      IF POS SF(bug1) # SF(bug2) THEN err($"invalid text selection")
                                                                     02001
      ELSE                                                           02002
         IF POS bug1 < bug2 THEN                                     02003
            FIND bug1 ^ptr1 bug2 > CH ^ptr2                          02004
         ELSE                                                        02005
            FIND bug2 ^tp1 bug1 > CH ^ptr2 tp1 ^ptr1;                02006
      RETURN;                                                        02007
      END.                                                           02008
                                                                     02009
   (ndr) PROCEDURE (bug, ptr1, ptr2); %verify number %               0985
      LOCAL sflg;                                                    0986
      REF bug, ptr1,ptr2;                                            0987
      sflg _ TRUE;                                                   0988
      IF NOT FIND bug < ($(D/ '-/ ',/ '-/ '+) ^ptr1)                 0989
      THEN err($"invalid number selection");                         0990
      IF NOT FIND ptr1 > ('-/'+/FR sflg) ((1$3D $(', 3D) ('. 1$D/-D)
      ^ptr2)/                                                        0991
         (1$D ('. 1$D/TRUE) ^ptr2)/('. 1$D ^ptr2))                   0992
      THEN err($"invalid number selection");                         0993
      FIND ptr2 < -D ^ptr2;                                          0994
      RETURN;                                                        0995
      END.
                                                                     0996
   (wdr) PROCEDURE (bug,ptr1,ptr2);                                  0997
      % This procedure delimits a "WORD" given a pointer to a
      character in a string.  The pointers PTR1 and PTR2 are set to
      the first and last characters, respectively, of the word.%
                                                                     02542
      REF bug, ptr1, ptr2;                                           0998
      FIND bug > CH $LD ^ptr2 bug < $LD ^ptr1;                       0999
      RETURN;                                                        01000
      END.                                                           01001
                                                                     01002
   %procedure nmdr has been moved to adrmnp, even though it logically
   belongs here.  this was done so it could be called while the OP or
   a compiler is in.  It gets called by xtrnam, lookup, %            03445
   %(nmdr) PROCEDURE (bug,ptr1,ptr2);                                03373
      %% statement name (and statement number) delimiter %%         03374
```

```
      REF bug, ptr1, ptr2;                                                  03375
      FIND bug >                                                            03376
         $(LD / '- /'' / '@ / '& / '( / ') ) ^ptr2                          03377
         bug < $(LD / '- /'' / '@ / '& / '( / ') ) ^ptr1;                   03378
         %% @ is alphabetic zero %%                                         03379
      RETURN;                                                               03380
      END. %                                                                03381
                                                                            03382
   (iddr) PROCEDURE (bug,ptr1,ptr2); %IDENT delimiter routine%              02640
      REF bug, ptr1, ptr2;                                                  02642
      FIND bug >                                                            02759
          $1('&/'^) $(LD / '- /'' ) ^ptr2                                   02762
         bug < $(LD / '- /'' ) ^ptr1;                                       02761
      RETURN;                                                               02647
      END.                                                                  02648
                                                                            02649
   (idIdr) PROCEDURE (bug1,bug2,ptr1,ptr2);                                 02661
      REF bug1, bug2, ptr1, ptr2;                                           02662
      LOCAL TEXT POINTER tp1, tp2;                                          02663
      IF POS SF(bug1) # SF(bug2) THEN err($"invalid IDENTLIST
      selection -- must be contained in one statement")                     02664
      ELSE                                                                  02665
         IF POS bug1 < bug2 THEN                                            02666
            FIND bug1 ^ptr1 bug2 > CH ^ptr2                                 02667
         ELSE                                                               02668
            FIND bug2 ^tp1 bug1 > CH ^ptr2 tp1 ^ptr1;                       02669
      iddr(&bug1, &ptr1, $tp2);                                             02678
      iddr(&bug2, $tp1, &ptr2);                                             02679
      RETURN;                                                               02670
      END.                                                                  02671
                                                                            02672
   (vdr) PROCEDURE (bug,ptr1,ptr2);                                         01013
      REF bug, ptr1, ptr2;                                                  01014
      FIND bug > CH $PT ^ptr2 bug < $PT ^ptr1;                              01015
      RETURN;                                                               01016
      END.                                                                  01017
                                                                            01018
   (findr) PROCEDURE (bug,ptr1,ptr2); %file name -- TENEX syntax%
                                                                            01019
      REF bug, ptr1, ptr2;                                                  01020
      FIND bug > $(LD/'./';/'>/'-) ^ptr2 bug < $(LD/'./';/'>/'-/'<)
      ^ptr1;                                                                01021
      RETURN;                                                               01022
      END.                                                                  01023
                                                                            01024
   (ldr) PROCEDURE (bug,ptr1,ptr2);                                         02130
      REF bug, ptr1, ptr2;                                                  02132
      FIND bug ^ptr1 ^ptr2;                                                 02139
      RETURN;                                                               02136
      END.                                                                  02137
                                                                            02138
%test and set bounds of structure%

                                                                            01033
   (plxset)                                                                 01034
      %Given a stid, this routine will return the stid's of the head
      and tail, respectively, of the plex in which the stid appears.%
```

```
                                                                    01035
      %-------------%                                               01036
      PROCEDURE(stid);                                              01037
      LOCAL grp1, %stid of head%                                    01038
         grp2; %stid of tail%                                       01039
      IF stid.stpsid = orgstid THEN grp1 _ grp2 _ stid              01040
      ELSE                                                          01041
         BEGIN                                                      01042
         grp1 _ gethed(stid);                                       01043
         grp2 _ getail(stid);                                       01044
         END;                                                       01045
      RETURN(grp1, grp2);                                           01046
      END.
                                                                    01047
% DAE PARSER %                                                     01128
   (getaddr) PROCEDURE(ptr); %read and evaluate address expression%
                                                                    04276
      %parameter: address of tpointer--starting tpointer for        04277
      relative addresses                                            04278
      Returns: after having updated the tpointer                    04279
      Error conditions: displays error message if receives unusual
      inpt                                                          04280
      %                                                             04281
      %----------------%                                           04282
      LOCAL mklink, da;                                             04284
      LOCAL TEXT POINTER oldptr, z1, z2;                            04285
      LOCAL STRING                                                  04286
         gadstr[200]; % to hold the entire string for the address %
                                                                    04287
      REF ptr, sysmsg, da;                                          04283
                                                                    04288
      (gad):                                                        04289
      ON SIGNAL ELSE IF nlmode = fulldisplay THEN rstlit();         04290
      slink _ cdlnk _ TRUE;                                         04291
      *gadstr* _ NULL;                                              04292
      gaddrlit($gadstr); %collect the literal%                      04293
      IF msjfn THEN  %monitoring commands%                          04294
         msrecord(maddrexp, 0, gadstr.L);                           04295
      FIND SF(*gadstr*) ^z1 SE(*gadstr*) ^z2;                       04296
      littolnk( FALSE, $z1, $z2);                                   04297
      oldptr _ ptr;                                                 04298
      oldptr[1] _ ptr[1];                                           04299
      FIND SF(*gadstr*) ^z1 SE(*gadstr*) ^z2;                       04300
      ON SIGNAL %catch any errors during interpretation%            04301
         =gaderr:                                                   04302
            BEGIN                                                   04303
            *[MESSAGE]*_SP,*[MESSAGE]*,"?  ";                       04304
            dismes(2, MESSAGE);                                     04305
            &sysmsg _ $"";                                          04306
            cueflg _ FALSE;                                         04307
            clrbuf(0);                                              04308
            ptr _ oldptr;                                           04309
            ptr[1] _ oldptr[1];                                     04310
            GOTO gad;                                               04311
            END;                                                    04312
         ELSE;                                                      04313
```

```
    &da _ lda();                                                    04317
    IF da.daempty AND NOT FIND SF(z1) $(SP/TAB) ( '( / '< / "--" )
    THEN err($"No file currently loaded.");                         04318
    cspvs _ caddexp($z1, $z2, &da, &ptr: cspvs[1], cspcacode,
    cspusqcod); %interpret the string%                              04314
    RETURN END.
                                                                    04315
(gaddrlit) PROCEDURE( % get a typed in address expression %         01552
    % FORMAL ARGUMENTS %                                           01553
       strptr);    % address of collection string %               01556
    % NORMAL RETURNS %                                             01557
       % none %                                                    01558
    % ABNORMAL RETURNS %                                          01559
       % a "popstate" SIGNAL is generated if BC or BW is typed and
       the literal string is empty %                               01560
    REF % VARIABLES %                                             01564
       strptr;                                                     01567
    % --------------- %                                           01569
    litreset _ FALSE;                                             01577
    % do prompting for text %                                     01583
       fbctl( incues, $"A:");                                     01584
    % look for a null literal %                                   01575
       IF lookc() = $ctln THEN                                    01576
          BEGIN                                                   01627
          inpt();                                                 01626
          RETURN;                                                 01629
          END;                                                    01628
    % read the literal %                                          01585
       IF nlmode = typewriter THEN                                01586
          echolt();                                               01590
       CASE rdlit( &strptr, 0) OF                                 01593
          = 3: % BC or BW typed and string already empty %        01594
             SIGNAL (popstate);                                   01595
          ENDCASE;                                                01596
       IF nlmode = fulldisplay AND littakedown THEN rstlit()      01597
       ELSE echoff();                                             01606
    RETURN;                                                       01621
    END.
                                                                  01622
(littolnk)          % turn a lit into a link maybe %              02010
    PROCEDURE                                                     02011
       (type,       % TRUE -> should be old/new filelink %        02012
       tp1,         % address of text pointer to start of lit %
                                                                  02013
       tp2);        % address of text pointer to end of lit %     02014
    LOCAL temp;                                                   02128
    REF tp1, tp2;                                                 02015
    temp _ type;                                                  02063
    FIND SF(tp1) $(SP/TAB);                                       02064
    IF FIND ( '( / '< / "--" ) THEN temp _ TRUE;                  02065
    LOOP                                                          02066
       CASE READC OF                                              02067
          = '), = '>, = ',, = ':                                  02068
             BEGIN                                                02069
             temp _ TRUE;                                         02070
             END;                                                 02071
```

```
            = '';                                                    02072
               CASE READC OF                                         02073
                  = ENDCHR:                                          02074
                     err($"Missing Character After Single Quote");
                                                                     02075
                  ENDCASE;                                           02076
            = '";                                                    02077
               LOOP                                                  02078
                  CASE READC OF                                      02079
                     = '": EXIT LOOP;                                02080
                     = ENDCHR:                                       02081
                        BEGIN                                        02082
                        ST tp1 _ SF(tp1) SE(tp1), '";                02083
                        tp2[1] _ tp2[1] + 1;                         02097
                        EXIT LOOP 2;                                 02084
                        END;                                         02085
                     ENDCASE;                                        02086
            = ENDCHR:  EXIT LOOP;                                    02087
            ENDCASE;                                                 02088
         IF temp OR type THEN                                        02089
            BEGIN                                                    02090
            IF NOT FIND SF(tp1) > $(SP/TAB) ( '( / '< / "--" ) THEN
                                                                     02091
               BEGIN                                                 02098
               ST tp1 _ '<, SF(tp1) SE(tp1);                         02092
               tp2[1] _ tp2[1] + 1;                                  02100
               END;                                                  02099
            FIND SE(tp1) $(SP/TAB);                                  02105
            CASE type OF                                             02102
               = TRUE:                                               02103
                  CASE READC OF                                      02112
                     = '), ='>: NULL;                                02113
                     = ',:                                           02114
                        BEGIN                                        02115
                        ST tp1 _ SF(tp1) SE(tp1), '>;                02116
                        tp2[1] _ tp2[1] + 1;                         02117
                        END;                                         02118
                     ENDCASE                                         02119
                        BEGIN                                        02120
                        ST tp1 _ SF(tp1) SE(tp1), ",>";              02121
                        tp2[1] _ tp2[1] + 2;                         02122
                        END;                                         02123
               ENDCASE                                               02104
                  CASE READC OF                                      02106
                     = '), ='>: NULL;                                02107
                     ENDCASE                                         02108
                        BEGIN                                        02109
                        ST tp1 _ SF(tp1) SE(tp1), '>;                02094
                        tp2[1] _ tp2[1] + 1;                         02111
                        END;                                         02110
            END;                                                     02095
         RETURN;                                                     02129
         END.

                                                                     02096
FINISH                                                               01536
```

SEQFIL

```
< NLS, SEQFIL.NLS.23, >, 6-Dec-77 12:46 JCP ;;;;
FILE seqfil % L10 <REL-NLS>Seqfil %% (L10,) (rel-nls,Seqfil.rel,) %
                                                                    02
%....declarations.....%
                                                                    03
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, p = 7, wa = 8, s = 9;   04
   REF prseqwk, tda;                                                05
   DECLARE                                                          06
      maxlev = 63, n40fst = 40B, tenfst = 0,                        07
      sfnfnm = 48, sfhpgn = 70,                                     08
      seqer = 4B8, seqeof = 1B9, maxtrc = 63, maxmclv = 1,          09
      ascmbk = 175B;
                                                                    010
   DECLARE                                                          011
      ok=1,          % successful return %                         012
      cat=0,         % catastrophic error return %                 013
      submission=1, % journal submission %                         014
      leaving=-1,    % file leaving system %                       015
      entering=0;    % file entering system %                      016
   DECLARE STRING formfeed = "
```

```
";                                                                  018
%.....output sequential, quickprint, and assembler.....%
                                                                    019
(outseq) PROCEDURE   % does the Output Sequential File command %
                                                                  02539
   (ofilnam,   % string containing name of the output file %      02540
   da,         % display area descriptor %                        02541
   forceup);   % whether alphas are to be forced upper case %    02542
   REF ofilnam, da;                                               02543
   dismes(1, $"Output Sequential in Progress");                   02544
   opseqf (&ofilnam, &da, forceup, FALSE, opsqfl);                02545
   dismes(0);                                                     02546
   RETURN;                                                        02547
   END.
                                                                  02548
(outqp)  PROCEDURE   % does the Output Quickprint command %      02549
   (ofilnam,   % string containing name of the output file %      02550
   da);        % display area descriptor %                        02551
   REF ofilnam, da;                                               02552
   dismes (1, $"Output Quickprint in Progress");                  02553
   opseqf (&ofilnam, &da, FALSE, TRUE, opqpfl);                   02554
   dismes(0);                                                     02555
   RETURN;                                                        02556
   END.
                                                                  02557
(outjm) PROCEDURE ( str, da ); % does Output Journal Mail command %
                                                                    039
   % FORMAL PARAMETERS %                                           040
      % str  - address of astring containing output file name %    041
      % address of display area %                                  042
   LOCAL pjb, pjba, pjbb, pjbc;                                   02073
      %may delete this statement and make pjb a formal parameter%
                                                                  02095
   LOCAL pjl, pjbrad, pjfile;                                     02094
   LOCAL pjbflag %journal branch found%, pjoflag %mail found%;    02081
   LOCAL pjcsp, pjrubmrk, pjcacode;                                 043
   LOCAL TEXT POINTER pjtp, tp1, tp2;                               044
   REF str, da, pjb;                                                045
   % set branch names, etc. %                                     02074
      pjbflag _ pjoflag _ 0;                                      02082
      %may delete these when pjb is made a calling parameter%     02096
         &pjb _ $pjba;                                            02075
         pjba_$"Journal"; pjbb_$"Info"; pjbc_$"Action";           02076
      pjl_3;                                                      02077
      pjfile _ da.dacsp.stfile;                                   02093
   WHILE (pjl_pjl-1) >= 0 DO IF (pjbrad _ pjb[pjl]) THEN          02078
      BEGIN                                                       02079
      % fetch stid for head of Journal branch %                     046
         b1 _ origin; % build stid %                               047
         b1.stfile _ pjfile; % for origin statemnt %               048
         b1[1] _ 1; % of currently loaded file %                   049
         specreg (pjbrad, nametyp, $b1); % fetch stid for Journal,
         Action, or Info branch %                                  050
         IF b1 = endfil THEN REPEAT LOOP; % there may not be any %
                                                                   051
         pjbflag _ TRUE;                                         02098
```

```
      IF getsub(b1) = b1 THEN REPEAT LOOP; %no mail here%     02099
      pjtp _ b1; % save stid for head %                         052
      pjtp[1] _ b1[1]; % of Journal branch %                    053
   IF NOT pjoflag THEN                                        02085
      BEGIN                                                   02083
      % save away current display area csp and viewspecs %      054
        pjusqc _ da.dausqcod; % save user seq gen addr %        055
        pjsavf _ da.davspec; % and viewspecs %                  056
        pjcsp _ da.dacsp; % save csp for restoration %          057
        pjrubmrk _ rubmrk; % save pointer to curr rubout flag %
                                                                058
        pjcacode _ da.dacacode;                                 059
      % trap signals so can retore the dispay area stuff %     060
        ON SIGNAL ELSE % can't afford to lose control %         061
          BEGIN                                                 062
          da.davspec _ pjsavf; % w/o restore user seq gen addr %
                                                                063
          da.dausqcod _ pjusqc; % and viewspecs %               064
          da.dacsp _ pjcsp; % restore csp %                     065
          da.dacacode _ pjcacode;                               066
          rubmrk _ pjrubmrk; % restore rubout flag pointer %    067
          END;                                                  068
      % set up display area stuff for new sequence %           069
        da.dausqcod _ $pjseqg; % instate Print Journal seq gen %
                                                                070
        da.davspec.vsusqf _ TRUE; % and enable it %             071
        da.davspec.vsbrof _ TRUE; % branch only %               072
        da.dacacode _ $pager; % content analyzer to put out form
        feeds %                                                 074
        da.davspec.vscapf _ TRUE; % turn on content analyzer %
                                                                075
        rubmrk _ $pjrubout; % intercept RUBOUT %                076
      % set flag so this won't be done again %                02086
        pjoflag _ TRUE;                                       02087
      dismes(1, $"Output of Journal Mail in progress" );        078
      END;                                                    02084
   % output the file %                                         077
      da.dacsp _ b1;                                            073
      opseqf( &str, &da, FALSE, TRUE, opqpfl );                 079
   END;                                                       02080
IF NOT pjbflag THEN err($" No Journal Branches");             02097
IF pjoflag THEN                                               02088
   BEGIN                                                      02089
   % restore the saved display area state stuff %              081
      dismes(0);                                                080
      da.davspec _ pjsavf; % restore user seq gen addr %        082
      da.dausqcod _ pjusqc; % and  viewspecs %                  083
      da.dacsp _ pjcsp; % restore csp %                         084
      da.dacacode _ pjcacode;                                   085
      rubmrk _ pjrubmrk; % restore rubout flag pointer %        086
   END                                                        02090
ELSE err($" You have no Journal mail");                       02092
RETURN; % return to caller %                                  087
END.
                                                              088
(pager) PROCEDURE ( sw );                                     089
```

```
   % FORMAL PARAMETERS %                                                  090
      % sw   address of the sequence generator work area %               091
   % TEXT POINTERS %                                                      092
      LOCAL TEXT POINTER  tp1;   % local copy of CCPOS %                  093
   REF sw;                                                               094
   % set tp1 to be the current CCPOS -- do not use CCPOS as it might
   be used by some routine downstream from us %                          095
      FIND ^tp1;                                                         096
   % look for a statment that looks like a journal citation %            097
      IF FIND tp1 ["Location:"] THEN                                     098
         BEGIN % journal citation, eject to a new page %                 099
         send(&sw,$formfeed);                                           0100
         END;                                                           0101
   RETURN(TRUE);                                                        0102
   END.
                                                                        0103
(outasm) PROCEDURE   % does the Output to Assembler File command %
                                                                        0104
   (ofilnam,   % string containing name of the output file %            0105
   da,         % display area descriptor %                              0106
   forceup);   % whether alphas are to be forced upper case %          0107
   REF ofilnam, da;                                                     0108
   dismes(1, $"Output for Assembler in Progress");                      0109
   opseqf (&ofilnam, &da, forceup, FALSE, opmcfl);                      0110
   dismes(0);                                                           0111
   RETURN;                                                              0112
   END.
                                                                        0113
(opseqf) PROCEDURE                                                     01666
   (ofilnam,   % string containing name of the output file %           01667
   da,         % display area descriptor %                             01668
   upflg,      % whether to force alphabetic characters uppercase %
                                                                       01669
   pgflg,      % whether to paginate %                                  01670
   typfil);    % type of file %
                                                                       02558
   %This is the main control file for generating a sequential file.
   Basically, it uses the sequence generator to get new statements
   and then adds these statements to a sequential file.  The file is
   both opened and closed here. %                                      01673
   %------------------%                                               01674
   LOCAL jfn, sw, toplev, seqstd, sv1, sv2, opsstate, lptflag;         01675
   LOCAL STRING string[200];                                           01676
   REF ofilnam, da, sw;                                                01677
                                                                       01678
   % open the output file %                                           01679
      ON SIGNAL ELSE opssig($opsstate, &sw, jfn);                      01680
      opsstate _ 0;                                                    03173
      IF (NOT oqapfg) OR ( FIND SF(*ofilnam*) "< *prtdir* "> ) THEN
                                                                       01681
         toplev _ gtjoosf                                              01784
      ELSE toplev _ 0;                                                 01682
      lptflag _ IF (FIND SF(*ofilnam*) "LPT:") THEN TRUE ELSE FALSE;
                                                                       03317
      IF NOT jfn _                                                     01683
         lgetjfn (0, &ofilnam, $txtext, toplev, $lit) THEN             01684
```

```
            SIGNAL (ofilerr, $lit);                                    01685
         opsstate _ 1;                                                 03174
         IF iptflag THEN oqapfg _ FALSE                                03318
         ELSE                                                          03321
            BEGIN                                                      03319
            !gtfdb( jfn, 1B6+1, $r3);                                  01686
            IF oqapfg THEN oqapfg _ IF r3.fdbnxf THEN FALSE ELSE TRUE;
                                                                       01687
            END;                                                       03320
         IF NOT sysopen (jfn, IF oqapfg THEN append ELSE write, chrtyp,
         $lit) THEN                                                    01688
            BEGIN                                                      01689
            reljfn (jfn);                                              01690
            SIGNAL (ofilerr, $lit);                                    01691
            END;                                                       01692
         opsstate _ 2;                                                 03175
      %initialize formatting stuff%                                    01693
         sfucf  _ upflg;                                               01694
         sfpgno _ pgflg;                                               01695
         sflnel _ CASE typfil OF                                       02927
            =opqpfl: IF uqpcolmax=0 THEN defqcolmax ELSE uqpcolmax;
                                                                       02928
            ENDCASE 72;                                                02929
         sfndlv _ da.daind/da.dahinc;                                  01697
         sfmxnd _ da.damind/da.dahinc;                                 01698
      %initialize sequence generator%                                  01699
         &sw _ openseq (seqstd _ da.dacsp, seqend(seqstd, sv1 _
         da.davspec, sv2 _ da.davspc2), sv1, sv2, da.dausqcod,
         da.dacacode);                                                 01700
      opsstate _ 3;                                                    03176
      %initialize stuff for doing pmap's%                              01701
         sfpmr1.LH _ 485;                                              01702
         sfpmr1.RH _ $sfbuff/1000B;                                    01703
         sfpmr2.LH _ jfn;                                              01704
         sfpmr2.RH _ 0;                                                01705
         sfpmr3 _ 140001B6;                                            01706
         sfbyte _ 0;                                                   01707
         stbufl _ slngth(                                              01708
            (sfbptr _ stbptr(empty) + $sfbuff),                        01709
            (sfndbf _ stbptr(5) + $sfbufe) );                          01710
      %initialize for output quickprint append%                        01711
      IF oqapfg AND (typfil = opqpfl) THEN                             01712
         BEGIN                                                         01713
         r1 _ jfn;                                                     01714
         r2 _ 'L - 100B;                                               01715
         !JSYS bout;                                                   01716
         END;                                                          01717
      opsstate _ 4;                                                    03177
      IF (IF typfil=opqpfl THEN &sw ELSE sfnxtst(&sw)) THEN            01718
         BEGIN                                                         01719
         IF sfpgno THEN                                                01720
            BEGIN                                                      01721
            bldsfhdr(&da);                                             01722
            *string* _ CR, LF, *sfhead*, '1, CR, LF, LF;               01723
            sflnpg _ 3;                                                01724
            CCPOS SF(*string*);                                        02262
```

```
        sftrln(stbptr(empty)+$string+1, stbptr(string.L)+$string+1);
                                                                    01725
        BUMP sfpgno;                                                01726
        END;                                                        01727
    CASE typfil OF                                                  02263
        = opmcfl:                                                   02264
            BEGIN                                                   01729
            IF (toplev _ sw.swclvl) = 0 THEN toplev _ 1;            01730
            sfmacpt(&da, &sw, toplev);                              01731
            END;                                                    01732
        = opqpfl: NULL;                                             02267
        ENDCASE sfputst(&da, &sw, typfil);                          02266
    WHILE sfnxtst(&sw) DO                                           01734
        IF typfil=opmcfl THEN sfmacpt(&da, &sw, toplev)             01735
        ELSE sfputst(&da, &sw, typfil);                             01736
    END;                                                            01737
%close file%                                                        01740
    IF NOT ogapfg THEN                                              01741
        BEGIN                                                       01742
        %pmap last page out%                                        01743
            r1 _ sfpmr1;                                            01744
            r2 _ sfpmr2;                                            01745
            r3 _ sfpmr3;                                            01746
            !JSYS pmap;                                             01747
        END                                                         01748
    ELSE                                                            01749
        BEGIN                                                       01750
        %sout last page out%                                        01751
            r1_sfpmr2.LH;                                           01752
            r2.RH_sfpmr1.RH*1000B;                                  01753
            r2.LH_-1;                                               01754
            r3_0;                                                   01755
            !JSYS sout;                                             01756
        END;                                                        01757
    opsstate _ 3;                                                   03179
    %dismiss page from fork%                                        01758
        r1 _ -1;                                                    01759
        r2 _ sfpmr1;                                                01760
        r3 _ sfpmr3;                                                01761
        !JSYS pmap;                                                 01762
    opsstate _ 2;                                                   03180
    % close sequence %                                             01738
        closeseq (&sw);                                             01739
    sfbyte _ sfbyte + slngth((stbptr(empty) + $sfbuff), sfbptr);
                                                                    01763
    r1.LH _ 12B;                                                    01764
    r1.RH _ jfn;                                                    01765
    r2 _ -1;                                                        01766
    r3 _ sfbyte;                                                    01767
    !JSYS chfdb;                                                    01768
    % close output file (but keep jfn) %                           01769
        IF NOT sysclose (jfn .V 4B11, $lit) THEN                    01770
            BEGIN                                                   01771
            reljfn (jfn);                                           01772
            dismes (2, $lit);                                       01773
            GOTO STATE;                                             01774
```

```
            END;                                                    01775
      opsstate _ 1;                                                 03181
      % delete old versions, if not Output Quickprint or Mailfile %
                                                                    01776
         IF typfil # opqpfl AND typfil # opmifl THEN delovsrns (jfn,
         nvtk);                                                     01777
      opsstate _ 0;                                                 03182
      % release the jfn %                                          01778
         reljfn (jfn);                                              01779
   RETURN;                                                          01780
   END.                                                             01781
                                                                    01782
(opssig) PROC(stflag, sw, jfn);                                     03184
   %signal routine for opseqf, close down stuff depending on stflag
   value.  the greater the value, the more stuff to do%             03209
   REF stflag, sw;                                                  03185
   LOOP CASE (stflag := stflag - 1) OF                              03186
      =4: !pmap(-1, sfpmr1, sfpmr3); %map work page out of addr.
      space%                                                        03192
      =3: closeseq(&sw);                                            03193
      =2:                                                           03194
         BEGIN %close file and release jfn%                         03195
         sysclose(jfn, $lit);                                       03196
         RETURN;                                                    03197
         END;                                                       03198
      =1:                                                           03200
         BEGIN                                                      03201
         reljfn(jfn);                                               03202
         RETURN;                                                    03203
         END;                                                       03204
      ENDCASE RETURN;                                               03205
   END.                                                             03206
                                                                    03208
(sfnxtst) %***% PROCEDURE (sw);                                     02336
   %This procedure calls the sequence generator to find the next
   statement.  If there are no more statements in the sequence, it
   returns FALSE.  Otherwise, it initializes CCPOS for reading the
   next statement, which may involve skipping the statement name. %
                                                                    02337
   %------------------%                                            02338
   LOCAL TEXT POINTER tptr;                                         02339
   REF sw;                                                          02340
   IF inptrf OR (tptr _ seqgen(&sw)) = endfil THEN                  02341
      RETURN(FALSE);                                                02342
   tptr[1] _ IF sw.swvspec.vsnamf THEN 1                            02343
      ELSE fchtxt(tptr);                                            02344
   CCPOS tptr;                                                      02345
   RETURN(TRUE);                                                    02346
   END.                                                             02347
                                                                    02348
(sfputst) PROCEDURE (da, sw, typfil);                               02968
   %This copies the contents of READC to a local string. It expects
   READC to be set up to begin reading.  When it completes processing
   this statement it performs the necessary cleanup to begin
   processing the next statement (by calling SFENDST) and returns.%
                                                                    02969
```

```
   %------------------%                                                02970
   LOCAL TEXT POINTER tptr;                                           02971
   LOCAL curlin, chrcnt, indcnt, begbp, bytptr, char, tabpos, count,
   inblnk, outblnk, frstblnk, cntrlstr, mrgnlvl, extraline, sigend,
   stnobp, prevchar, string[50];                                      02972
   LOCAL STRING stnsig[30];                                           02973
   REF da, sw;                                                        02974
   %initialize this statement%                                        02975
      IF sw.swvspec.vsindf OR sw.swvspec.vsrind THEN                  02976
         BEGIN                                                        02977
         IF sw.swvspec.vsrind AND (sw.swvspec.vsplxf OR
         sw.swvspec.vsbrof) THEN mrgnlvl _ sw.swslvl                  02978
            ELSE mrgnlvl _ 1;                                         02979
         indcnt _ MAX (0, MIN (sfndlv * (sw.swclvl-mrgnlvl),
         sfmxnd));                                                    02980
         END                                                          02981
      ELSE indcnt _ 0;                                                02982
      bytptr _ begbp _ stbptr(empty) + $string;                       02983
      IF (chrcnt _ indcnt) > 0 THEN                                   02984
         FOR count _ 1 UP UNTIL > indcnt DO ^bytptr _ SP;             02985
      IF sw.swvspec.vsstnf AND NOT sw.swvspec.vsstnr                  02986
      AND sw.swcstid.stpsid # origin AND NOT sw.swstid.stastr THEN
                                                                      02987
         BEGIN % statement number on left %                           02988
         *stnsig* _ NULL;                                             02989
         IF sw.swvspec.vssidf                                         02990
            THEN % display sid's %                                    02991
               *stnsig* _ '0, STRING( getsid( sw.swcstid ) )          02992
            ELSE % display line numbers%                              02993
               fechnm (sw.swsvw, $stnsig);                            02994
         FOR count _ empty + 1 UP UNTIL > stnsig.L DO                 02995
            ^bytptr _ *stnsig*[count];                                02996
         ^bytptr _ SP;                                                02997
         chrcnt _ chrcnt + stnsig.L + 1;                              02998
         END;                                                         02999
      curlin _ 1;                                                     03000
      extraline _ FALSE;                                              03001
   LOOP   % for each line %                                           03002
      BEGIN                                                           03003
      inblnk _ CCPOS;                                                 03004
      outblnk _ frstblnk _ bytptr;                                    03005
      LOOP                                                            03006
         CASE char _ READC OF                                         03007
            = SP:                                                     03008
               BEGIN                                                  03009
               prevchar _ CCPOS;                                      03010
               IF (chrcnt _ chrcnt + 1) > sflnel THEN EXIT LOOP 1;
                                                                      03011
               ^bytptr _ SP;                                          03012
               outblnk _ bytptr;                                      03013
               inblnk _ CCPOS;                                        03014
               END;                                                   03015
            = ENDCHR: EXIT LOOP 2;                                    03016
            = TAB:                                                    03017
               BEGIN                                                  03018
               tabpos _ MIN (sflnel+1,                                03019
```

```
            fndtab(&da,chrcnt + (IF typfil = opqpfl THEN 0 ELSE
            1)) - 1);                                          03020
         FOR chrcnt UP UNTIL >= tabpos DO                      03021
            ^bytptr _ SP;                                      03022
         IF chrcnt > sflnel+1 THEN EXIT LOOP 1;                03023
         outblnk _ bytptr;                                     03024
         inblnk  _ CCPOS;                                      03025
         END;                                                  03026
      = CR, =EOL:  EXIT LOOP 1;                                03027
      < 40B:  %handle control chars for output quickprint%
                                                               03028
         IF typfil NOT= opqpfl THEN GO TO eccntrl              03029
         ELSE  %put in printing strings for control characters%
                                                               03030
            BEGIN                                              03031
                                                               03032
            %force page eject on ^L & don't print anything%
                                                               03033
            IF (char = 'L - 100B) AND sfpgno THEN              03034
               BEGIN                                           03035
               sflnpg _ 200; % checked in sftrlu at end of line
               %                                               03036
               GO TO eccntrl;                                  03037
               END;                                            03038
                                                               03039
            %get printing string and put in output string%
                                                               03040
            cntrlstr _ npstrad(char);                          03041
            IF chrcnt + [cntrlstr].L <= sflnel+1 THEN          03042
               BEGIN   %stash away printing string if it fits%
                                                               03043
               FOR count _ empty + 1 UP UNTIL > [cntrlstr].L DO
                                                               03044
                  BEGIN                                        03045
                  ^bytptr _ *[cntrlstr]*[count];               03046
                  chrcnt _ chrcnt + 1;                         03047
                  END;                                         03048
               chrcnt _ chrcnt - 1;                            03049
               END                                             03050
            ELSE chrcnt _ chrcnt + [cntrlstr].L - 1;           03051
                                                               03052
            %continue as if normal characters%                 03053
            GO TO eccntrl;                                     03054
            END;                                               03055
      ENDCASE                                                  03056
         BEGIN                                                 03057
         (eccntrl):                                            03058
            IF (chrcnt _ chrcnt + 1) > sflnel THEN             03059
               BEGIN                                           03060
               IF frstblnk # outblnk THEN %there are blanks%
                                                               03061
                  BEGIN                                        03062
                  bytptr _ outblnk;                            03063
                  %reset readwk%                               03064
                     tptr _ sw.swstid;                         03065
                     tptr[1] _ inblnk;                         03066
```

```
                        CCPOS tptr;                                     03067
                     END                                                03068
                  ELSE    %no blanks; back up 1 char%                   03069
                     BEGIN                                              03070
                     tptr _ sw.swstid;                                  03071
                     tptr[1] _ prevchar;                                03072
                     CCPOS tptr;                                        03073
                     END;                                               03074
                  EXIT LOOP 1;                                          03075
                  END;                                                  03076
               IF char > 40B THEN                                       03077
                  BEGIN                                                 03078
                  ^bytptr _ IF sfucf AND char IN ['a, 'z] THEN          03079
                  char - 40B ELSE char;                                 03080
                  END                                                   03081
               ELSE IF typfil # opqpfl THEN ^bytptr _ char;             03082
               prevchar _ CCPOS;                                        03083
               END;                                                     03084
      IF curlin >= sw.swvspec.vstrnc THEN EXIT LOOP 1;                  03085
      IF char = SP AND chrcnt > sflnel THEN                             03086
         %Read chars; if SP or TAB, write on this line, not the next%   03087
         LOOP                                                           03088
            BEGIN                                                       03089
            prevchar _ CCPOS;                                           03090
            CASE (char _ READC) OF                                      03091
               =SP, =TAB:  NULL;  %throw them away%                     03092
               =CR, =EOL:  EXIT LOOP 1;                                 03093
               =ENDCHR:  EXIT LOOP 2;                                   03094
               ENDCASE                                                  03095
                  BEGIN      %Back up and pretend we never read it%     03096
                  tptr _ sw.swstid;                                     03097
                  tptr[1] _ prevchar;                                   03098
                  CCPOS tptr;                                           03099
                  EXIT LOOP 1;                                          03100
                  END;                                                  03101
            END;                                                        03102
      ^bytptr _  CR;                                                    03103
      ^bytptr _  LF;                                                    03104
      sttrln (begbp, bytptr);   % transfer line to output buffer %     03105
      % initialize for next line %                                      03106
         BUMP curlin;                                                   03107
         bytptr _ begbp;                                                03108
         IF (chrcnt _ indcnt) > 0 THEN                                  03109
            FOR count _ 1 UP UNTIL > indcnt DO ^bytptr _ SP;             03110
   END;                                                                 03111
% end of statement stuff %                                            03112
IF sw.swvspec.vsstnf AND sw.swvspec.vsstnr                              03113
AND sw.swcstid.stpsid # origin AND NOT sw.swstid.stastr THEN            03114
   BEGIN  %statement number on right%                                   03115
   *stnsig* _ NULL;                                                     03116
   IF sw.swvspec.vssidf                                                 03117
      THEN % display sid's %                                            03118
         *stnsig* _ '0, STRING( getsid( sw.swcstid ) )                  03119
```

```
      ELSE % display line numbers%                                    03120
         fechnm (sw.swsvw, $stnsig);                                  03121
      IF chrcnt + 1 + stnsig.L > sflnel THEN                          03122
         BEGIN %must go to new line%                                  03123
         ^bytptr _ CR;                                                03124
         ^bytptr _ LF;                                                03125
         sftrln (begbp, bytptr);   % transfer line to output buffer %
                                                                      03126
         bytptr _ begbp;                                              03127
         chrcnt _ 0;                                                  03128
         extraline _ TRUE;                                            03129
         END                                                          03130
      ELSE extraline _ FALSE;                                         03131
      FOR count _ chrcnt + 1 UP UNTIL > sflnel - stnsig.L DO          03132
         ^bytptr _ SP;                                                03133
      FOR count _ empty + 1 UP UNTIL > stnsig.L DO                    03134
         ^bytptr _ *stnsig*[count];                                   03135
      END;                                                            03136
   IF NOT (sw.swvspec.vsidtf AND extraline) THEN                      03137
      BEGIN  %only if signatures on and go on separate line%          03138
      ^bytptr _ CR;                                                   03139
      ^bytptr _ LF;                                                   03140
      sftrln (begbp, bytptr);   % transfer line to output buffer %
                                                                      03141
      sigend _ sflnel;                                                03142
      END                                                             03143
   ELSE                                                               03144
      BEGIN                                                           03145
      sigend _ sflnel - stnsig.L - 3;                                 03146
      stnobp _ bytptr;                                                03147
      END;                                                            03148
   IF sw.swvspec.vsidtf THEN                                          03149
      BEGIN                                                           03150
      bytptr _ begbp;                                                 03151
      *stnsig* _ NULL;                                                03152
      fechsig (sw.swcstid, $stnsig);                                  03153
      FOR count _ 1 UP UNTIL > sigend - stnsig.L DO                   03154
         ^bytptr _ SP;                                                03155
      FOR count _ empty + 1 UP UNTIL > stnsig.L DO                    03156
         ^bytptr _ *stnsig*[count];                                   03157
      IF extraline THEN bytptr _ stnobp;                              03158
      ^bytptr _ CR;                                                   03159
      ^bytptr _ LF;                                                   03160
      sftrln (begbp, bytptr);   % transfer line to output buffer %
                                                                      03161
      END                                                             03162
   ELSE IF sw.swvspec.vsblkf AND NOT extraline THEN                   03163
      BEGIN                                                           03164
      bytptr _ begbp;                                                 03165
      ^bytptr _ CR;                                                   03166
      ^bytptr _ LF;                                                   03167
      sftrln (begbp, bytptr);   % transfer line to output buffer %
                                                                      03168
      END;                                                            03169
   RETURN;                                                            03170
   END.
```

```
                                                                    03171
(sfmacpt) PROCEDURE (da, sw, toplev);                               0416
   %This copies the contents of READC to a local string, for output
   to the assembler. It works like sfputst, except it inserts tabs at
   the beginning of statements that are of a lower level than the
   first statement in the series. In addition a statement consisting
   of only one space will be output as a null line. %               0417
   %------------------%                                             0418
                                                                    0419
   LOCAL char, bytptr, count, begptr, tmpptr, string[400];          0420
   REF da, sw;                                                      0421
                                                                    0422
   % initialization %                                               0423
   begptr _ bytptr _ tmpptr _ stbptr(empty) + $string;              0424
                                                                    0425
   % put out semi-colon in front of origin statement %              0426
   IF (NOT sw.swstid.stastr) AND (sw.swstid.stpsid = origin) THEN
                                                                    0427
      ^bytptr _ '; ;                                                0428
                                                                    0429
   %  put out leading tabs in front of first line of statement if
   appropriate and indenting is turned on  %                        0430
   IF (count _ sw.swclvl - toplev) < 0  THEN count _ 0;             0431
   IF count > 0 AND da.davspec.vsindf THEN                          0432
      DO ^bytptr _ TAB  UNTIL (count _ count - 1) <= 0;             0433
   tmpptr _ bytptr;                                                 0434
                                                                    0435
   LOOP                                                             0436
      BEGIN                                                         0437
      LOOP                                                          0438
         CASE  char _ READC  OF                                     0439
            = ENDCHR:  EXIT LOOP 1;                                 0440
            = CR, =EOL:                                             0441
               BEGIN                                                0442
               ^bytptr _ CR;                                        0443
               ^bytptr _ LF;                                        0444
               EXIT LOOP 1;                                         0445
               END;                                                 0446
            ENDCASE                                                 0447
               BEGIN                                                0448
               % convert to upper case if appropriate %             0449
               ^bytptr _ IF sfucf AND char IN ['a, 'z] THEN         0450
                  char - 40B ELSE char;                             0451
               END;                                                 0452
                                                                    0453
      % put out null statement if appropriate %                     0454
      IF char = ENDCHR AND ^tmpptr = SP AND slngth(tmpptr,bytptr) = 0
                                                                    0455
         THEN bytptr _ begptr;                                      0456
                                                                    0457
      CASE char OF                                                  0458
         = ENDCHR: EXIT LOOP 1;                                     0459
         = EOL, = CR: NULL;                                         0460
         ENDCASE                                                    0461
            BEGIN                                                   0462
            ^bytptr _ CR;                                           0463
```

```
            ^bytptr _ LF;                                          0464
            END;                                                   0465
         sftrln(begptr, bytptr);  % transfer line to output buffer %
                                                                   0466
                                                                   0467
         % reinitialize for going thru loop %                      0468
         begptr _ tmpptr _ bytptr _stbptr(empty) + $string;        0469
         END;                                                      0470
                                                                   0471
      ^bytptr _ CR;                                                0472
      ^bytptr _ LF;                                                0473
      sftrln(begptr, bytptr);  % transfer line to output buffer %  0474
      RETURN;                                                      0475
      END.                                                         0476
                                                                   0477
(sftrln) PROCEDURE (begbp, bytptr);                                0478
      %This routine copies characters bounded by begbp and bytptr (both
      byte pointers) into sfbuff and pmaps sfbuff into the print file
      when the buffer is full.%                                    0479
      %------------------%                                        0480
      LOCAL STRING string[100];                                    0481
      UNTIL begbp = bytptr DO                                      0482
         BEGIN                                                     0483
         ^sfbptr _ ^begbp;                                         0484
         IF sfbptr = sfndbf THEN                                   0485
            BEGIN                                                  0486
            IF NOT oqapfg THEN                                     0487
               BEGIN                                               0488
               %pmap page out%                                     0489
                  r1 _ sfpmr1;                                     0490
                  r2 _ sfpmr2;                                     0491
                  r3 _ sfpmr3;                                     0492
                  !JSYS pmap;                                      0493
               END                                                 0494
            ELSE                                                   0495
               BEGIN                                               0496
               %sout page out%                                     0497
                  r1_sfpmr2.LH;                                    0498
                  r2.RH_sfpmr1.RH*1000B;                           0499
                  r2.LH_-1;                                        0500
                  r3_-5000B;                                       0501
                  !JSYS sout;                                      0502
               END;                                                0503
            %dismiss page from fork%                               0504
               r1 _ -1;                                            0505
               r2 _ sfpmr1;                                        0506
               r3 _ sfpmr3;                                        0507
               !JSYS pmap;                                         0508
            sfbptr _ stbptr(empty) + $sfbuff;                      0509
            BUMP sfpmr2;                                           0510
            sfbyte _ sfbyte + sfbufl;                              0511
            END;                                                   0512
         END;                                                      0513
      IF sfpgno AND (sflnpg _ sflnpg + 1) >= 60 THEN               0514
         BEGIN                                                     0515
         *string* _ 14B, CR, LF, *sfhead*, STRING(sfpgno), CR, LF, LF;
```

```
                                                                    0516
      sflnpg _ 3;                                                   0517
      sftrln (stbptr(empty)+$string+1, stbptr(string.L) + $string+1);
                                                                    0518
      BUMP sfpgno;                                                  0519
      END;                                                          0520
   RETURN;                                                          0521
   END.                                                             0522
                                                                    0523
(bldsfhdr) PROCEDURE (da);                                          0524
   %This routine builds a page header for output quickprint.%       0525
   %------------------%                                            0526
   LOCAL vspwrd, count;                                             0527
   LOCAL STRING filstr[100];                                        0528
   REF da;                                                          0529
   IF oqnhfg THEN                                                  02256
      BEGIN                                                        02257
      %No header wanted; page numbers only%                        02258
         *sfhead* _ "
         Page ";              %57 spaces%                          02259
         RETURN;                                                   02260
      END;                                                         02261
   *filstr* _ NULL;                                                 0530
   vspwrd _ da.davspec;                                             0531
   %put initials into sfhead%                                       0532
      *sfhead* _ "   ", *initsr*, ",   ";                          0533
   getdat($sfhead);                                                 0534
   %put truncation and level values into sfhead%                    0535
      IF vspwrd.vstrnc < maxtrc OR vspwrd.vslev < maxlev THEN       0536
         BEGIN                                                      0537
         IF vspwrd.vstrnc < maxtrc THEN                             0538
            *sfhead* _ *sfhead*, " T=", STRING(vspwrd.vstrnc), ',
                                                                    0539
         ELSE *sfhead* _ *sfhead*, " T=ALL,";                       0540
         IF vspwrd.vslev < maxlev THEN                              0541
            *sfhead* _ *sfhead*, " L=", STRING(vspwrd.vslev), ',    0542
         ELSE *sfhead* _ *sfhead*, " L=ALL,";                       0543
         END;                                                       0544
   %get file name%                                                  0545
      filnam(da.dacsp.stfile, $filstr);                             0546
   %blank fill to position for file name%                           0547
      FOR count _ 1 UP UNTIL > MAX (1, sfhpgn - filstr.L - sfhead.L)
      DO                                                           01783
         *sfhead* _ *sfhead*, SP;                                   0548
   %put file name in header%                                        0549
      *sfhead* _ *sfhead*, *filstr*;                                0550
   %2 blanks to position for page number%                           0551
      *sfhead* _ *sfhead*, "  ";                                   0552
   RETURN;                                                          0553
   END.                                                             0554
                                                                    0555
%.....insert sequential.....%
                                                                    0556
(inseq) PROCEDURE   % does the Copy Sequential command %           0557
   (stid,     % statement after which to insert %                   0558
   rlevcnt,   % level relative to stid %                            0559
```

```
    ifilnam,    % string containing the name of the input file %    0560
    filtyp);    % type of input %                                   0561
                                                                    0562
    % This is the main control routine for insert sequential.  The
    input file is opened and closed here. %                         0563
    %------------------%                                           0564
    LOCAL count, levblk, lstblk, curblk, curdpth, levchg, char, pos,
    offset, jfn, t1, t2, t3, t4, t5, t6, term, firststid;           0565
    LOCAL TEXT POINTER start, end;                                  0566
    REF ifilnam;                                                    0567
                                                                    0568
    dismes(1, $"Copy Sequential in Progress");                      0569
    firststid_0;                                                   02282
    % open the input file %                                         0570
       IF NOT jfn _ lgetjfn (0, &ifilnam, $txtext, gtjoisf, $lit) THEN
                                                                    0571
          SIGNAL (ofilerr, $lit);                                   0572
       IF NOT sysopen (jfn, read, chrtyp, $lit) THEN                0573
          BEGIN                                                     0574
          reljfn (jfn);                                             0575
          SIGNAL (ofilerr, $lit);                                   0576
          END;                                                      0577
    IF filtyp = assfil THEN GOTO inseqaws;                          0578
    levblk _ lstblk _ curdpth _ 0;                                  0579
    offset _ IF filtyp = n40fil THEN n40fst ELSE tenfst;            0580
    *lit* _ NULL;                                                   0581
    isread (jfn, $sar, CR);                                         0582
    pos _ empty + 1;                                                0583
    WHILE *sar*[pos] = LF DO BUMP pos;                              0584
    IF filtyp # macfil THEN                                         0585
       CASE *sar*[pos]  OF                                          0586
          =ascmbk:                                                  0587
             levblk _ lstblk _ *sar*[pos _ pos + 1] - offset;       0588
          =SP, =TAB:                                                0589
             BEGIN                                                  0590
             levblk _ 1;                                            0591
             LOOP                                                   0592
                CASE *sar*[pos _ pos + 1] OF                        0593
                   =SP, =TAB: BUMP levblk;                          0594
                   ENDCASE EXIT LOOP;                               0595
             lstblk _ levblk;                                       0596
             pos _ pos - 1;                                         0597
             END;                                                   0598
          ENDCASE pos _ pos - 1                                     0599
    ELSE pos _ pos - 1;                                             0600
    curdpth _ 0;                                                    0601
    LOOP                                                            0602
       BEGIN                                                        0603
       UNTIL (pos := pos + 1) > sar.L DO                            0604
          CASE char _ *sar*[pos] OF                                 0605
             =ascmbk:                                               0606
                BEGIN                                               0607
                count _ *sar*[pos _ pos + 1] - offset;              0608
                UNTIL (count := count - 1) = 0 DO                   0609
                *lit* _ *lit*, SP;                                  0610
                END;                                                0611
```

```
        = LF, = 0: NULL;                                                0612
        = ENDCHR: IF lit.L NOT= empty THEN GOTO insql1 ELSE GOTO
        insql2;                                                         0613
        = EOL, =CR: %end of statement%                                  0614
          BEGIN                                                         0615
          (insql1):                                                     0616
            FIND SF(*lit*) ^start SE(*lit*) ^end;                       0617
            stid _ <STRMNP, cinssta> (stid, rlevcnt, $start,
            $end);                                                      0618
            IF firststid=0 THEN firststid_stid;                        02283
          (insql2): *lit* _ NULL;                                       0619
          IF NOT isread(jfn, $sar, CR) THEN EXIT LOOP 2;                0620
          pos _ empty + 1;                                              0621
          WHILE *sar*[pos] = LF DO BUMP pos;                            0622
          IF filtyp # macfil THEN                                       0623
            CASE *sar*[pos] OF                                          0624
              %determine level of next statement%                       0625
              =ascmbk:                                                  0626
                curblk _ *sar*[pos _ pos + 1] - offset;                 0627
              =SP, =TAB:                                                0628
                BEGIN                                                   0629
                count _ 1;                                              0630
                LOOP                                                    0631
                  CASE *sar*[pos _ pos + 1] OF                          0632
                    =SP, =TAB:                                          0633
                      BEGIN                                             0634
                      IF filtyp = macfil AND count >=
                      maxmclv THEN EXIT LOOP;                           0635
                      BUMP count;                                       0636
                      END;                                              0637
                    ENDCASE EXIT LOOP;                                  0638
                pos _ pos - 1;                                          0639
                curblk _ count;                                         0640
                END;                                                    0641
              ENDCASE                                                   0642
                BEGIN                                                   0643
                curblk _ 0;                                             0644
                pos _ pos - 1;                                          0645
                END                                                     0646
          ELSE                                                          0647
            BEGIN                                                       0648
            curblk _ 0;                                                 0649
            pos _ pos - 1;                                              0650
            END;                                                        0651
          IF curblk THEN                                                0652
            BEGIN                                                       0653
            IF NOT levblk THEN levblk _ curblk;                         0654
            IF filtyp # macfil THEN                                     0655
              CASE lstblk OF                                            0656
                =curblk: %same level%                                   0657
                  rlevcnt _ levsuc;                                     0658
                < curblk: %levdown a level%                             0659
                  IF (curblk - lstblk) >= levblk                        0660
                  OR curblk/levblk > curdpth THEN                       0661
                    BEGIN                                               0662
                    rlevcnt _ levdown;                                  0663
```

```
                        BUMP curdpth;                                   0664
                        END                                             0665
                     ELSE rlevcnt _ levsuc;                             0666
                  ENDCASE %up a level%                                  0667
                     BEGIN                                              0668
                     levchg _ (lstblk - curblk) / levblk;               0669
                     IF levchg < 1 AND curdpth > 2 THEN                 0670
                        levchg _ (curdpth - 1) - curblk/levblk;
                                                                        0671
                     IF levchg > curdpth THEN levchg _ curdpth;
                                                                        0672
                     curdpth _ curdpth - levchg;                        0673
                     rlevcnt _ levsuc;                                  0674
                     WHILE (levchg _ levchg - 1) >= 0 DO                0675
                        rlevcnt _ rlevcnt + 1;                          0676
                     END;                                               0677
               END                                                      0678
            ELSE                                                        0679
               BEGIN                                                    0680
               rlevcnt _ levsuc;                                        0681
               IF lstblk THEN                                           0682
                  WHILE (curdpth := curdpth - 1) > 0 DO                 0683
                     rlevcnt _ rlevcnt + 1;                             0684
               curdpth _ 0;                                             0685
               END;                                                     0686
            lstblk _ curblk;                                            0687
            END;                                                        0688
         ENDCASE *lit* _ *lit*, char;                                   0689
   IF NOT isread(jfn, $sar, CR) THEN EXIT LOOP;                         0690
   pos _ empty;                                                         0691
   END;                                                                 0692
IF lit.L NOT= empty THEN                                                0693
   BEGIN                                                                0694
   FIND SF(*lit*) ^start SE(*lit*) ^end;                                0695
   stid _ <STRMNP, cinssta> (stid, rlevcnt, $start, $end);              0696
   END;                                                                 0697
%close file%                                                           0698
   (inseqcl):                                                           0699
   IF NOT sysclose (jfn, $lit) THEN                                     0700
      BEGIN                                                             0701
      reljfn (jfn);                                                     0702
      err($lit);                                                        0703
      END;                                                              0704
dismes(0);                                                              0705
IF firststid=0 THEN firststid_stid;                                    02284
RETURN(firststid);                                                      0706
                                                                        0707
(inseqaws):                                                             0708
                                                                        0709
% the following code is concerned with insert sequential assembler
file with structure.  the first statment in the assembler file is
inserted without modification at the level specified by the user.
succeeding statements are placed one level levdown for each tab at
the front of the statement and up the appropriate number of levels
if a succeeding statement has less tabs than the current
statement.  ASCIZ, ASCII, and SIXBIT statements are handled as
```

```
follows:  the first non-blank character following the delimiter
character for the ASCIZ, ASCII, and SIXBIT pseudo-ops serves as a
delimiter.  all characters between this delimiter and its next
occurrence are inserted literally in the statement, i.e., EOL or
CR do not serve to delimit the statement, and no level adjustments
are made. %                                                      0710
                                                                 0711
% insert the first statement %                                   0712
*lit* _ NULL;                                                    0713
IF NOT isread(jfn, $sar, CR) THEN GOTO iswsend;                  0714
pos _ empty + 1;                                                 0715
WHILE *sar*[pos] = LF  DO BUMP pos;                              0716
curdpth _ 0;                                                     0717
LOOP                                                             0718
   CASE char _ *sar*[pos := pos + 1]  OF                         0719
      = LF, = 0: NULL;                                           0720
      = ENDCHR:  % the only way we get here is end of file %     0721
         BEGIN                                                   0722
         IF lit.L NOT= empty THEN                                0723
            BEGIN                                                0724
            FIND SF(*lit*) ^start SE(*lit*) ^end;                0725
            cinssta(stid, rlevcnt, $start, $end);                0726
            END;                                                 0727
         GOTO inseqcl;  % go close the file %                    0728
         END;                                                    0729
      = EOL, =CR:                                                0730
         BEGIN                                                   0731
         FIND SF(*lit*) ^start SE(*lit*) ^end;                   0732
         stid _ cinssta(stid, rlevcnt, $start, $end);            0733
         EXIT LOOP;                                              0734
         END;                                                    0735
      ENDCASE *lit* _ *lit*, char;                               0736
                                                                 0737
LOOP   % this is the main loop for the rest of the statements %
                                                                 0738
   BEGIN                                                         0739
                                                                 0740
   % initialization %                                           0741
   *lit* _ NULL;                                                 0742
   rlevcnt _ levsuc;                                             0743
                                                                 0744
   % read in a statement from the assembler file %               0745
   IF NOT isread(jfn, $sar, CR) THEN EXIT LOOP;                  0746
      % i.e. GOTO iswsend %                                      0747
   pos _ empty + 1;                                              0748
                                                                 0749
   % bypass LF %                                                 0750
   WHILE *sar*[pos] = LF DO BUMP pos;                            0751
                                                                 0752
   % find out level of statement relative to previous statement %
                                                                 0753
   count _ 0;                                                    0754
   WHILE *sar*[pos] = TAB DO                                     0755
      BEGIN                                                      0756
      BUMP count;                                                0757
      BUMP pos;                                                  0758
```

```
      END;                                                              0759
   CASE count OF                                                        0760
      > curdpth:                                                        0761
         BEGIN                                                          0762
         rlevcnt _ levdown;                                             0763
         count _ count - curdpth;                                       0764
         curdpth _ curdpth + 1;                                         0765
         WHILE (count _ count - 1) > 0 DO *lit* _ *lit*, TAB;           0766
         END;                                                           0767
      < curdpth:                                                        0768
         BEGIN                                                          0769
         t1 _ curdpth - count;                                          0770
         WHILE (t1 := t1 - 1 ) > 0 DO rlevcnt _ rlevcnt +1;             0771
         curdpth _ count;                                               0772
         END;                                                           0773
      ENDCASE;                                                          0774
                                                                        0775
   % we are now past any leading tabs and will gobble up the rest
   of the statement one character at a time %                           0776
   LOOP                                                                 0777
      CASE char _ *sar*[pos := pos + 1] OF                              0778
         = LF, = 0: NULL;                                               0779
         = ENDCHR:  % the only way we get here is end of file %
                                                                        0780
            BEGIN                                                       0781
            IF lit.L NOT= empty THEN                                    0782
               BEGIN                                                    0783
               FIND SF(*lit*) ^start SE(*lit*) ^end;                    0784
               cinssta(stid, rlevcnt, $start, $end);                    0785
               END;                                                     0786
            GOTO inseqcl;  % go close the file %                        0787
            END;                                                        0788
         = EOL, = CR:                                                   0789
            BEGIN                                                       0790
            FIND SF(*lit*) ^start SE(*lit*) ^end;                       0791
            stid _ cinssta(stid, rlevcnt, $start, $end);                0792
            EXIT LOOP;                                                  0793
            END;                                                        0794
         = 'A, ='a:   % check for asciz or ascii %                      0795
            BEGIN                                                       0796
            *lit* _ *lit*, char;                                        0797
            IF pos + 4 <= sar.L THEN                                    0798
               BEGIN                                                    0799
               t1 _ *sar*[pos + 0];                                     0800
               t2 _ *sar*[pos + 1];                                     0801
               t3 _ *sar*[pos + 2];                                     0802
               t4 _ *sar*[pos + 3];                                     0803
               t5 _ *sar*[pos + 4];                                     0804
               IF ( (t1 = 'S) OR (t1 = 's) ) AND                        0805
                  ( (t2 = 'C) OR (t2 = 'c) ) AND                        0806
                  ( (t3 = 'I) OR (t3 = 'i) ) AND                        0807
                  ( (t4 = 'Z) OR (t4 = 'z) OR                           0808
                    (t4 = 'I) OR (t4 = 'i) ) AND                        0809
                  ( (t5 NOT IN ['A, 'Z]) AND                            0810
                    (t5 NOT IN ['a, 'z]) AND                            0811
                    (t5 # '$) AND (t5 # '%) AND                         0812
```

```
  (t5 # '.) AND (t5 # LF) AND                                    0813
  (t5 # EOL) AND (t5 # CR) )  THEN                               0814
BEGIN                                                            0815
*lit* _ *lit*, t1, t2, t3, t4, t5;                               0816
pos _ pos + 4;                                                   0817
IF (t5 # SP) AND (t5 # TAB) THEN term _ t5                       0818
ELSE LOOP                                                        0819
   IF (pos _ pos + 1) <= sar.L THEN                              0820
      IF ((term _ *sar*[pos]) = SP) OR                           0821
         (term = TAB) THEN *lit* _ *lit*, term
                                                                 0822
      ELSE                                                       0823
         BEGIN                                                   0824
         *lit* _ *lit*, term;                                    0825
         EXIT LOOP;                                              0826
         END                                                     0827
   ELSE                                                          0828
      BEGIN                                                      0829
      IF NOT isread(jfn, $sar, CR) THEN                          0830
         GOTO iswsend;                                           0831
      pos _ empty;                                               0832
      END;                                                       0833
                                                                 0834
% we have now gobbled up asciz or ascii and the
first delimiter and are gobbling succeeding
characters until we find the second occurrence
of the delimiter %                                               0835
LOOP                                                             0836
   CASE char _ *sar*[pos _ pos + 1] OF                           0837
      = LF, = 0: NULL;                                           0838
                                                                 0839
      % the only way we get here is end of file%
                                                                 0840
      = ENDCHR:                                                  0841
         BEGIN                                                   0842
         IF lit.L NOT= empty THEN                                0843
            BEGIN                                                0844
            FIND SF(*lit*) ^start SE(*lit*)
            ^end;                                                0845
            cinssta(stid, rlevcnt, $start,
            $end);                                               0846
            END;                                                 0847
         GOTO inseqcl;  % go close the file %
                                                                 0848
         END;                                                    0849
      = EOL, = CR:                                               0850
         BEGIN                                                   0851
         *lit* _ *lit*, EOL;                                     0852
         IF NOT isread(jfn, $sar, CR) THEN                       0853
            GOTO iswsend;                                        0854
         pos _ empty;                                            0855
         END;                                                    0856
      = term:                                                    0857
         BEGIN                                                   0858
         *lit* _ *lit*, char;                                    0859
         pos _ pos + 1;                                          0860
```

```
                        EXIT LOOP;                                      0861
                        END;                                            0862
                     ENDCASE *lit* _ *lit*, char;                       0863
                  END;                                                  0864
               END;                                                     0865
            END;                                                        0866
         = 'S, ='s:   % check for sixbit %                              0867
            BEGIN                                                       0868
            *lit* _ *lit*, char;                                        0869
            IF pos + 5 <= sar.L THEN                                    0870
               BEGIN                                                    0871
               t1 _ *sar*[pos + 0];                                     0872
               t2 _ *sar*[pos + 1];                                     0873
               t3 _ *sar*[pos + 2];                                     0874
               t4 _ *sar*[pos + 3];                                     0875
               t5 _ *sar*[pos + 4];                                     0876
               t6 _ *sar*[pos + 5];                                     0877
               IF ( (t1 = 'I) OR (t1 = 'i) ) AND                        0878
                  ( (t2 = 'X) OR (t2 = 'x) ) AND                        0879
                  ( (t3 = 'B) OR (t3 = 'b) ) AND                        0880
                  ( (t4 = 'I) OR (t4 = 'i) ) AND                        0881
                  ( (t5 = 'T) OR (t4 = 't) ) AND                        0882
                  ( (t6 NOT IN ['A, 'Z]) AND                            0883
                    (t6 NOT IN ['a, 'z]) AND                            0884
                    (t6 # '$) AND (t6 # '%) AND                         0885
                    (t6 # '.) AND (t6 # LF) AND                         0886
                    (t6 # EOL) AND (t6 # CR) ) THEN                     0887
                  BEGIN                                                 0888
                  *lit* _ *lit*, t1, t2, t3, t4, t5, t6;                0889
                  pos _ pos + 5;                                        0890
                  IF (t6 # SP) AND (t6 # TAB) THEN term _ t6            0891
                  ELSE LOOP                                             0892
                     IF (pos _ pos + 1) <= sar.L THEN                   0893
                        IF ((term _ *sar*[pos]) = SP) OR                0894
                           (term = TAB) THEN *lit* _ *lit*, term
                                                                        0895
                        ELSE                                            0896
                           BEGIN                                        0897
                           *lit* _ *lit*, term;                         0898
                           EXIT LOOP;                                   0899
                           END                                          0900
                     ELSE                                               0901
                        BEGIN                                           0902
                        IF NOT isread(jfn, $sar, CR) THEN               0903
                           GOTO iswsend;                                0904
                        pos _ empty;                                    0905
                        END;                                            0906
                                                                        0907
                  % we have now gobbled up sixbit and the first
                  delimiter and are gobbling succeeding characters
                  until we find the second occurrence of the
                  delimiter %                                           0908
                  LOOP                                                  0909
                     CASE char _ *sar*[pos _ pos + 1] OF                0910
                        = LF, = 0: NULL;                                0911
                                                                        0912
```

```
                        % the only way we get here is end of file%
                                                                    0913
                        = ENDCHR:                                   0914
                           BEGIN                                    0915
                           IF lit.L NOT= empty THEN                 0916
                              BEGIN                                 0917
                              FIND SF(*lit*) ^start SE(*lit*)
                              ^end;                                 0918
                              cinssta(stid, rlevcnt, $start,
                              $end);                                0919
                              END;                                  0920
                           GOTO inseqcl;  % go close the file %
                                                                    0921
                           END;                                     0922
                        = EOL, = CR:                                0923
                           BEGIN                                    0924
                           *lit* _ *lit*, EOL;                      0925
                           IF NOT isread(jfn, $sar, CR) THEN        0926
                              GOTO iswsend;                         0927
                           pos _ empty;                             0928
                           END;                                     0929
                        = term:                                     0930
                           BEGIN                                    0931
                           *lit* _ *lit*, char;                     0932
                           pos _ pos + 1;                           0933
                           EXIT LOOP;                               0934
                           END;                                     0935
                        ENDCASE *lit* _ *lit*, char;                0936
                     END;                                           0937
                  END;                                              0938
               END;                                                 0939
            ENDCASE *lit* _ *lit*, char;                            0940
      END;                                                          0941
                                                                    0942
   (iswsend):   % cleanup at end of file %                          0943
   t1 _ sar.L;                                                      0944
   pos _ empty;                                                     0945
   WHILE (t1 := t1 - 1) > 0 DO *lit* _ *lit*, *sar*[pos _ pos + 1];
                                                                    0946
   IF lit.L NOT= empty THEN                                         0947
      BEGIN                                                         0948
      FIND SF(*lit*) ^start SE(*lit*) ^end;                         0949
      cinssta(stid, rlevcnt, $start, $end);                         0950
      END;                                                          0951
   GOTO inseqcl;   % close the file and return %                    0952
                                                                    0953
   END.                                                             0954
                                                                    0955
(isread)                                                            0956
   %This procedure is used during insert sequential to read another
   character from the input file.  It expects the input jfn, the
   address of a string, and a terminating character, as arguments, in
   that order.  It puts the characters read into the string; it
   returns true until it has an empty input buffer and encounters an
   EOF.%                                                            0957
   %------------------%                                            0958
```

```
   PROCEDURE(jfn, inbuff, lstchr);                                          0959
   LOCAL error, maxcnt, char, count, bytptr;                                0960
   REF inbuff;                                                              0961
   bytptr _ chbmty + &inbuff;                                               0962
   maxcnt _ inbuff.M;                                                       0963
   count _ empty;                                                           0964
   LOOP                                                                     0965
      BEGIN                                                                 0966
      !bin( jfn );                                                          0967
      IF NOT (char _ r2 .A 377B) % null byte % THEN                         0968
         BEGIN                                                              0969
         !gtsts( jfn );                                                     0970
         error _ r2;                                                        0971
         IF error .A seqer # 0 THEN% Input error %                          0972
            BEGIN                                                           0973
            IF NOT sysclose (jfn, $lit) THEN                                0974
               BEGIN                                                        0975
               reljfn (jfn);                                                0976
               err ($lit);                                                  0977
               END                                                          0978
            ELSE err ($"Input File Error");                                 0979
            END;                                                            0980
         IF error .A seqeof # 0 THEN % EOF read-- check if buffer
         empty %                                                            0981
            BEGIN                                                           0982
            IF (inbuff.L _ count) > empty THEN RETURN(TRUE)                 0983
            ELSE RETURN( FALSE );                                           0984
            END;                                                            0985
         END;                                                               0986
      ^bytptr _ char;                                                       0987
      IF ((count _ count + 1) >= maxcnt) OR (char = lstchr) OR (char
      = EOL AND lstchr = CR) THEN                                           0988
         BEGIN                                                              0989
         inbuff.L _ count;                                                  0990
         RETURN(TRUE);                                                      0991
         END;                                                               0992
      END;                                                                  0993
   END.
                                                                            0994
(inseqn) PROCEDURE   %does the input sequential command (new way):
double CR means end of st.%                                                 02368
   (stid,       % statement after which to insert                             %
                                                                            02369
    rlevcnt,    % level relative to stid                                     %
                                                                            02370
    ifilnam,    % string containing name of input file                       %
                                                                            02371
    just);      % were multiple spaces added to right justify?              02372
   ------------------------------------------------------------------------%
                                                                            02373
   % declarations %                                                         02374
      LOCAL jfn, place, dummy, indent1, indent2, level, adj, done ;
                                                                            02375
      LOCAL TEXT POINTER stop, end, t1, t2, z1, z2 ;                        02376
      LOCAL STRING line1[999], line2[999], st[2000], string[20] ;
                                                                            02377
```

```
      REF ifilnam ;                                                    02378
      level _ done _ 0;                                                02379
   dismes (1, $"Insert Sequential in Progress ");                      02380
   % handle different types %                                          02381
   % open input file %                                                 02382
      IF NOT jfn _ lgetjfn (0, &ifilnam, $txtext, gtjoisf, $lit)
                                                                       02383
         THEN SIGNAL (ofilerr, $lit) ;                                 02384
      IF NOT sysopen (jfn, read, chrtyp, $lit) THEN                    02385
         BEGIN                                                         02386
         reljfn (jfn) ;                                                02387
         SIGNAL (ofilerr, $lit) ;                                      02388
         END;                                                          02389
   % insert dummy statement %                                          02390
      *string* _ "dummy";                                              02391
      FIND SF(*string*) ^t1 SE(*string*) ^t2;                          02392
      dummy _ place _ cinssta (stid, rlevcnt, $t1, $t2) ;              02393
   % for all data in input file %                                      02394
      LOOP %each cycle a statement%                                    02395
         BEGIN                                                         02396
         adj _ indent1 _ indent2 _ 0 ;                                 02397
         % get first line %                                            02398
            DO %skip blank lines%                                      02399
               BEGIN                                                   02400
               indent1 _ inseql (jfn, $line1, just) ;                  02401
               IF indent1 = -1 THEN                                    02402
                  BEGIN                                                02403
                  done _ 1;                                            02404
                  EXIT LOOP;                                           02405
                  END;                                                 02406
               END                                                     02407
            UNTIL line1.L>0 ;                                          02408
         % get second line %                                           02409
            indent2 _ inseql (jfn, $line2, just) ;                     02410
            IF indent2 = -1 THEN                                       02411
               done _ 1;                                               02412
         % add both lines to *st* %                                    02413
            *st* _ *line1*, *line2* ;                                  02414
         % figure out level %                                          02415
            IF (line2.L AND indent2) > 0 THEN indent1 _ indent2;  %
            we go by the second line in all cases except where there
            is no second line or it is not indented.  Then we go by
            the first line. %                                          02416
            CASE indent1 OF                                            02417
               > levind[level]:                                        02418
                  IF level = 11 THEN                                   02419
                     adj _ levsuc                                      02420
                  ELSE                                                 02421
                     BEGIN                                             02422
                     BUMP level;                                       02423
                     levind[level] _ indent1 ;                         02424
                     IF (place := getsub(place)) = place               02425
                        THEN adj _ levdown                             02426
                        ELSE                                           02427
                           BEGIN                                       02428
                           place _ getail(place);                      02429
```

```
                        adj _ levsuc;                                   02430
                        END;                                            02431
                    END;                                                02432
                < levind[level]:                                        02433
                    BEGIN                                               02434
                    BUMP DOWN level ;                                   02435
                    place _ getup (place) ;                             02436
                    REPEAT CASE;                                        02437
                    END;                                                02438
                ENDCASE  % =levind[level] %                             02439
                    adj _ levsuc ;                                      02440
            % add lines to *st* until double EOL %                      02441
              IF line2.L > 0 THEN                                       02442
                LOOP                                                    02443
                    BEGIN                                               02444
                    IF inseql(jfn,$line1,just) = -1 THEN                02445
                        BEGIN                                           02446
                        done _ 1;                                       02447
                        EXIT LOOP;                                      02448
                        END;                                            02449
                    IF line1.L=0 THEN EXIT LOOP;                        02450
                    IF (st.L + line1.L) > st.M THEN                     02451
                        BEGIN  %start new statement%                    02452
                        FIND SF(*st*) ^z1 SE(*st*) ^z2;                 02453
                        place _ cinssta (place, adj, $z1, $z2) ;        02454
                        adj _ levsuc ;                                  02455
                        st.L _ 0 ;                                      02456
                        END;                                            02457
                    *st* _ *st*, *line1* ;                              02458
                    END;                                                02459
            % insert the statement %                                    02460
              FIND SE(*st*) ^end $NP ^stop > ;                          02461
              ST stop end _ NULL ;                                      02462
              FIND SF(*st*) ^z1 SE(*st*) ^z2;                           02463
              place _ cinssta (place, adj, $z1, $z2) ;                  02464
              IF done THEN EXIT LOOP;                                   02465
            END;                                                        02466
        % close the input file %                                        02467
          IF NOT sysclose (jfn, $lit) THEN                              02468
            BEGIN                                                       02469
            reljfn (jfn) ;                                              02470
            dismes (2, $lit) ;                                          02471
            GOTO STATE;                                                 02472
            END;                                                        02473
        % delete dummy statement %                                      02474
          IF (place _ getsub(dummy)) # dummy THEN                       02475
            cmovgro (dummy, levsuc, gethed(place) , getail(place),
            FALSE, 0) ;                                                 02476
          place _ getsuc(dummy);                                        02477
          cdelsta (dummy, FALSE, 0) ;                                   02478
        dismes(0) ;                                                     02479
        RETURN(place);                                                  02480
        END.                                                            02481
(inseql) PROCEDURE (jfn, line, justified) ; %gets a line for inseqn%
                                                                        02482
        % declarations %                                               02483
```

```
    LOCAL indent, short ;                                                  02484
    LOCAL TEXT POINTER pt1, pt2 ;                                          02485
    REF line ;                                                             02486
% get line %                                                              02487
    line.L _ 0 ;                                                           02488
    IF NOT isread (jfn, &line, CR) THEN RETURN (-1); %couldn't get
    line%                                                                  02489
    short _ (IF line.L < 50 THEN TRUE ELSE FALSE) ;                        02490
% if within 8 chars of RM, and if fully justified, remove extra
spaces if less than or equal to 5 in a row %                               02491
    IF justified AND NOT short THEN                                        02492
      BEGIN                                                                02493
      FIND SF(*line*) $NP ;                                                02494
      LOOP                                                                 02495
        BEGIN                                                              02496
        IF NOT FIND [2SP] THEN EXIT LOOP;                                  02497
        IF FIND ^pt1 _pt1 $3SP ^pt2 PT THEN                                02498
          ST pt1 pt2 _ NULL                                                02499
        ELSE FIND $SP;                                                     02500
        END;                                                               02501
      END;                                                                 02502
% count and delete leading spaces %                                        02503
    CCPOS SF(*line*) ;                                                     02504
    indent _ 0;                                                            02505
    CASE READC OF                                                          02506
      =SP:                                                                 02507
        BEGIN                                                              02508
        indent _ indent + 1 ;                                              02509
        REPEAT CASE ;                                                      02510
        END;                                                               02511
      =TAB:                                                                02512
        BEGIN                                                              02513
        indent _ indent + 8 ;                                              02514
        REPEAT CASE ;                                                      02515
        END;                                                               02516
      =NP:  REPEAT CASE;                                                   02517
      ENDCASE FIND ^pt1 _pt1;                                              02518
% delete trailing spaces %                                                02519
    CCPOS SE(*line*) ;                                                     02520
    CASE READC OF                                                          02521
      =EOL, =CR:                                                           02522
        IF short THEN                                                      02523
          BEGIN                                                            02524
          FIND ^pt2 >;                                                     02525
          *line* _ pt1 pt2, EOL ;                                          02526
          IF line.L=1 THEN line.L _ 0 ;                                    02527
          END                                                              02528
        ELSE REPEAT CASE ;                                                 02529
      =NP:  REPEAT CASE;                                                   02530
      ENDCASE                                                              02531
        IF line.L < line.M THEN                                            02532
          BEGIN                                                            02533
          FIND ^pt2 _pt2 >;                                                02534
          *line* _ pt1 pt2, SP ;                                           02535
          END;                                                             02536
RETURN (indent);                                                           02537
```

```
   END.                                                                 02538
(hiop) PROCEDURE (acs); % NLS Host Input/Output Processor %             01164
   % Declarations %                                                     01165
      LOCAL                                                            01166
         direction, % file entering/leaving system %                   01167
         parmchr,   % single char of parms %                           01168
         i,         % working index variable %                         01169
         injfn,     % source file jfn %                                01170
         outjfn,    % result file jfn %                                01171
         bp1, bp2;  % byte pointers %                                  01172
      LOCAL TEXT POINTER                                               01173
         tp1, tp2, tp3, tp4, z1, z2; % working pointers %              01174
      LOCAL STRING                                                     01175
         parms [40],     % parameters %                                01176
         tail [40],      % tail of parms %                            01177
         infile [40],    % source filename %                          01178
         outfile [40],   % result filename %                          01179
         fieldname [40], % name of field %                            01180
         fieldvalue [40],% value of field %                           01181
         viewspecs [40], % viewspecs string %                         01182
         addr [40],      % statement addr %                           01183
         convtype [40],  % conversion algorithm %                     01184
         errstr [40];    % error string %                             01185
      REF acs; % AC's upon entry to NLS %                              01186
   % Save direction and JFN %                                          01187
      ON SIGNAL ELSE                                                   01188
         hiortn (cat, $"Conversion setup error: ", $sysmsg, $""); %
         catch signals %                                               01189
      r1 _ acs [7]; % get xwd direction, input jfn %                   01190
      ! HLRE 2,1; % isolate direction, propagate sign %                01191
      ! HRRZS 1; % clean input jfn %                                   01192
      injfn _ r1; % save input jfn %                                   01193
      direction _ r2; %save direction %                                01194
   % Build parameter string %                                          01195
      bp1 _ chbptr(empty) + $parms;                                    01196
      bp2 _ chbptr(empty) + &acs + 7;                                  01197
      r1 _ bp1; % load destination AC %                                01198
      r2 _ bp2; % load source AC %                                     01199
      r3 _ 0; % terminate on NUL %                                     01200
      ! JSYS sout; % copy the ASCIZ string %                          01201
      bp2 _ r1; % save dest-end pointer %                              01202
      parms.L _ slngth (bp1, bp2); % set parms length %               01203
   % Init defaults %                                                  01204
      *viewspecs* _ ""; % null viewspecs %                            01205
      *convtype* _ "S"; % convert to/from sequential file %           01206
      *addr* _ ".1"; % point to top of file %                         01207
   % Extract fields from parameter string %                           01208
      CCPOS *parms*; % init text pointer %                             01209
      *tail* _ *parms*; % init tail of parms %                        01210
      IF parms.L # 0 THEN % skip parse if null parms %                 01211
         LOOP                                                          01212
            BEGIN % here for each field of parms %                     01213
            IF NOT FIND ^tp1 $LD ^tp2 ': ^tp3 ((['',] ^tp4 _tp4) /
            (SE(tp3) ^tp4 >)) THEN hiortn (cat, $"What is '", $tail,
            $"'?"); % extract next field and its value %               01214
            *fieldname* _ + tp1 tp2; % save name of vield %            01215
```

```
            *fieldvalue* _ tp3 tp4; % and its value %                    01216
            IF fieldname.L # 1 THEN GOTO badfield; % all field types
            are one char %                                               01217
            CASE *fieldname* [1] OF % save field value appropriately
            %                                                            01218
               ='T: *convtype* _ + SF(*fieldvalue*) SE(*fieldvalue*);
               % conversion type %                                       01219
               ='V: *viewspecs* _ *fieldvalue*; % viewspecs %            01220
               ='N: *addr* _ *fieldvalue*; % statement %                 01221
               ENDCASE (badfield): hiortn (cat, $fieldname, $"-field
               undefined", $""); % undefined field type %                01222
            *tail* _ tp4 SE(tp4); % parms all processed? %               01223
            IF tail.L = 0 THEN EXIT LOOP; % quit if so %                 01224
            END;                                                         01225
      % Load source file %                                              01226
         ON SIGNAL ELSE                                                  01227
            hiortn (cat, $"Can't load file: ", $sysmsg, $""); % catch
            signals %                                                    01228
         jfnstr (injfn, $infile); % fetch filename %                     01229
         tda.dacsp _ orgstid;  tda.daccnt _ 1;                           01230
         tda.dacsp.stfile _ cloafil ($infile); % load source file %
                                                                         01231
      % Prepare output file %                                           01232
         ON SIGNAL ELSE                                                  01233
            hiortn (cat, $"Can't get output file: ", $sysmsg, $""); %
            catch signals %                                              01234
         IF NOT outjfn _ sgtjfn (6B11, $"HIOPWORK.TXT", $errstr) THEN
         hiortn (cat, $"Problems with output file: ", $errstr, $""); %
         abort if problems with output file %                            01235
         jfnstr (outjfn, $outfile); % fetch output filename %            01236
      % Verify and instate viewspecs %                                  01237
         ON SIGNAL ELSE                                                  01238
            hiortn (cat, $viewspecs, $" are invalid viewspecs: ",
            $sysmsg); % catch signals %                                  01239
         feedlt (&tda, $viewspecs); % build viewspecs %                  01240
      % Verify and position to statement addr %                         01241
         ON SIGNAL ELSE                                                  01242
            hiortn (cat, $addr, $" is an invalid addr: ", $sysmsg); %
            catch signals %                                              01243
         b1 _ tda.dacsp; % get current... %                              01244
         b1 [1] _ tda.daccnt; % ...command marker %                      01245
         FIND SF(*addr*) ^z1 SE(*addr*) ^z2;                             01246
         caddexp ($z1, $z2, &tda, $b1); % interpret TNLS addr %          01247
         tda.dacsp _ b1;  tda.daccnt _ b1[1];                            01248
      % Invoke the conversion %                                         01249
         ON SIGNAL ELSE                                                  01250
            hiortn (cat, $convtype, $"-conversion error: ", $sysmsg); %
            catch signals %                                              01251
         IF direction # leaving THEN hiortn (cat, $"Input not yet
         supported", $"", $""); % file input not yet implemented %       01252
         CASE *convtype* [1] OF                                          01253
            ='A: outasm ($outfile, &tda, FALSE);                         01254
            ='P: coutproc ($outfile, &tda, opprdv, 0);                   01255
            ='Q: outqp ($outfile, &tda);                                 01256
            ='S: outseq ($outfile, &tda, FALSE);                         01257
            ENDCASE hiortn (cat, $convtype, $"-conversion type
```

```
      undefined", $"");                                              01258
   hiortn (outjfn, $convtype, $"-conversion", $""); % successful
   completion %                                                      01259
   END.                                                              01260
%.....output device and output compiler.....%
                                                                     01261
%.....processor dispatch.....%
                                                                     01262
(processor) PROCEDURE   % start and run a processor %               01939
   (prcnam,  % string containing the name of the processor %        01940
   tp,       % text pointer to first thing to feed the processor %
                                                                     01941
   da,       % display area descriptor of the input to processor %
                                                                     01942
   type,     % type of the processor %                              01943
   outdsg,   % output designator:  either the address of a buffer or
   the jfn of an output file %                                       01944
   prcjfn);  %JFN of processor if no processor name pased or 0%     01945
                                                                     01946
      % ****** The Catalog System has its own version of PROCESSOR so
      please tell Walt if you make any changes ****** %              01947
                                                                     01948
                                                                     01949
   LOCAL stid, size, error, savpreg, savwareg, vspec, sdb, subsbn;
                                                                     01950
   LOCAL TEXT POINTER fchartp, tptr;                                 01951
   LOCAL STRING ssysnam[40];                                         01952
   REF prcnam, tp, da, sdb;                                          01953
                                                                     01954
   % set up to get first character for the processor %               01955
      vspec _ da.davspec;                                            01956
      IF type = cmptyp THEN vspec.vsstnf _ TRUE;                     01957
         % turn statement numbers on for MPL %                       01958
      &prseqwk  _ openseq (tp, seqend(tp, vspec, da.davspc2), vspec,
      da.davspc2, da.dausqcod, IF tp.stastr THEN 0 ELSE da.dacacode);
                                                                     01959
      IF (stid _ seqgen(&prseqwk)) = endfil THEN                     01960
         BEGIN                                                       01961
         closeseq (&prseqwk);                                        01962
         SIGNAL (prcerr, $"No processor input");                     01963
         END;                                                        01964
      IF tp.stastr THEN FIND (tp ^fchartp)                           01965
      ELSE FIND (SF(stid) ^fchartp);                                 01966
   IF NOT prcjfn AND NOT (prcjfn _ lgetjfn($subdir, &prcnam, $savext,
   gtjprf, $lit))                                                    01967
      THEN SIGNAL (prcerr, $lit);                                    01968
   % get a string containing name of the processor %                 01969
      IF type = upgtyp THEN *ssysnam* _ "NLSL10"                     01970
      ELSE                                                           01971
         BEGIN                                                       01972
         jfntostr (prcjfn, $ssysnam, 11B9);                          01973
         IF NOT FIND SF(*ssysnam*) "SUBSYS" ^tptr THEN               01974
            *ssysnam* _ "PRVPRC"                                     01975
         ELSE *ssysnam* _ tptr SE(*ssysnam*);                        01976
         END;                                                        01977
   %map out high pages%                                              01978
```

```
   hmapout();                                                    01979
%get and save current subsystem name%                            03283
   !getnm();                                                     03284
   subsbn _ r1;                                                  03285
% set the subsystem name %                                       01980
   r1 _ getsbn ($ssysnam);                                       01981
   !JSYS setnm;                                                  01982
% "get" the processor %                                          01983
   r1.LH _ 4B5;                                                  01984
   r1.RH _ prcjfn;                                               01985
   !JSYS get;                                                    01986
IF [prcadr + 1] THEN chntab[9] _ [prcadr + 1] .V 1B6;            01987
savpreg _ p; savwareg _ wa;  % save wa and p registers since the
compilers and the Output Processor will clobber them %           01988
ON SIGNAL ELSE GOTO procrestore;                                 01989
CASE type OF                                                     01990
   = upgtyp:   % compile a user program %                        01991
      BEGIN                                                      01992
      IF NOT cppflag THEN dismes (1, $"Compiling User Program");
                                                                 01993
      swork _ fchartp; swork[1] _ fchartp[1]; fechc1(1, $swork);
                                                                 01994
      %swork[2] contains byte count+1, swork[4] has byte pointer%
                                                                 01995
      error _ [[prcadr]]                                         01996
         (-1, $prgetst, outdsg, , $lev1lc, swork[4], swork[2]-1
         :size);                                                 01997
      END;                                                       01998
   = cmptyp:   % compile a program into an output file %         01999
      BEGIN                                                      02000
      dismes (1, $"Compiling");                                  02001
      swork _ fchartp; swork[1] _ fchartp[1]; fechc1(1, $swork);
                                                                 02002
      %swork[2] contains byte count+1, swork[4] has byte pointer%
                                                                 02003
      error _ [[prcadr]]                                         02004
         (outdsg, $prgetst, prseqwk.swsvw, , $lev1lc, swork[4],
         swork[2]-1);                                            02005
      END;                                                       02006
   = odtyp:    % output device type %                            02007
      BEGIN                                                      02008
      dismes (1, IF exp                                          02009
         THEN $"Experimental Output Processor"                   02010
         ELSE $"Processing Output");                             02011
      IF NOT stid.stastr THEN                                    02314
         BEGIN                                                   02315
         IF NOT lodprop( stid, txttyp : &sdb) THEN               02362
            err($"No text block associated with node");          02363
         opbp _ opcbp _ &sdb + sdbhdl + 4407B8;                  02317
         opcmax _ sdb.schars + 1;                                02318
         END                                                     02319
      ELSE                                                       02320
         BEGIN                                                   02321
         opbp _ opcbp _ chbmty + stid.stpsid;;                   02322
         opcmax _ [stid.stpsid].L + 1;                           02323
         END;                                                    02324
```

```
        opccpos _ 1;                                                    02015
        opnewst _ TRUE;                                                 02016
        error _ [[prcadr]] ($oprwrk, $lev1lc, opbp, opcmax);            02017
        END;                                                            02018
      ENDCASE   % some other type of processor %                        02019
        BEGIN                                                           02020
        dismes (1, $"Processor in progress");                           02021
        swork _ fchartp; swork[1] _ fchartp[1]; fechc1(1, $swork);      02022
        %swork[2] contains byte count+1, swork[4] has byte pointer%     02023
        error _ [[prcadr]] (outdsg, $prgetst, , , $lev1lc, swork[4],   02024
        swork[2]-1);                                                    02025
        END;                                                            02026
    (procrestore):                                                      02027
    % restore wa and p registers %                                     02028
      p _ savpreg; wa _ savwareg;                                       02029
    %restore stack overflow pseudo-interrupt%                           02030
      chntab[9] _ $stkovr .V 1B6;                                       02031
    %get nls pages back%                                                02032
      hmapin();                                                         02033
    ON SIGNAL ELSE                                                      02034
      BEGIN                                                             02035
      ON SIGNAL ELSE; % In case closeseq generates a signal %           02036
      %close the sequence%                                              02037
        closeseq (&prseqwk);                                            02038
      inptrf _ 0;                                                       02039
      END;                                                              02040
    %restore subsystem name to NLS%                                     02041
      r1 _ subsbn;                                                      02042
      !JSYS setnm;                                                      02043
    %take message away%                                                 02044
      dismes (0);                                                       02045
    %close the sequence%                                                02046
      closeseq (&prseqwk);                                              02047
    %reset entry vector back to NLS%                                    02048
      setvec();                                                         02049
    IF inptrf THEN %^O pseudo-interrupt%                                02050
      BEGIN                                                             02051
      inptrf _ 0;                                                       02052
      clrbuf (1);                                                       02053
      IF type # upgtyp THEN                                             02054
        IF NOT <IOEXEC, sysclose> (outdsg, $lit) THEN                   02055
          dismes (2, $lit);                                             02056
      SIGNAL(-4, $"User Terminated Process");                           02057
      END;                                                              02058
    RETURN (error, size);                                               02059
    END.

(mapped) RECORD % pmap handle and page status. %                        01587
    thand[26], % handle from handle. %                                  01588
    hndacc[10]; % access information. %                                 01589
(hmapout)PROC;                                                          01590
    LOCAL count, fbase, handle;                                         01591
    %Keep track of pages mapped out%                                    01592
      FOR count _ (fbase _ $bfree/1000B) UP UNTIL >= $efree/1000B DO
```

```
                                                                    01593
        BEGIN                                                       01594
        r1.LH _ 4B5;                                                01595
        r1.RH _ count;                                              01596
        !JSYS rpacs;                                                01597
        IF SKIP !TLNN r2, 10000B THEN %page exists%                 01598
           BEGIN                                                    01599
           %get handle for page%                                    01600
              r3 _ r2;                                              03312
                 %save access--don't trust TOPS20 RMAP to give
                 correct access%                                    03314
                 %known TOPS20 bug as of 12-APR-77--JDH%            03315
              !JSYS rmap;                                           01601
              r2 _ r3; %restore access from RPACS%                  03313
              IF r1.LH = 4B5 THEN pmfmap(TRUE);                     03286
                 %the pmf stuff doesn't work with tops20 so we have
                 to hndle it%                                       03287
              handle _ r1;                                          01602
              handle.hndacc _ r2.hndacc;                            01603
           %remove page%                                            01604
              r2.LH _ 4B5;                                          01605
              r2.RH _ count;                                        01606
              r1 _ -1;                                              01607
              r3 _ 0;                                               01608
              !JSYS pmap;                                           01609
           END                                                      01610
        ELSE  %page does not exist%                                 01611
           handle _ -1;                                             01612
        freemap[count-fbase] _ handle;                              01613
        END;                                                        01614
   RETURN;                                                          01615
   END.

                                                                    01616
(pmfmap)PROCEDURE(removf);                                          03288
   %map fork page in r1 to "UPMF" file and return new handle in r1.
   r2 contains access, returns it in tact.%                         03289
   LOCAL STRING filnms[100], errstr[200];                           03290
   !PUSH s,r1; !PUSH s,r2;                                          03293
   IF NOT pmfjfn THEN %must get the file%                           03291
      BEGIN                                                         03292
      *filnms* _ '<, *userstr*, '>, "$UPMF$.PGS;P707000;T", 0;      03294
      IF (pmfjfn _ sgtjfn(0, $filnms, $errstr)) AND NOT SKIP
      !openf(pmfjfn, 302000B) THEN reljfn(pmfjfn:=0);               03295
      IF NOT pmfjfn THEN err($"Can't open PMF file");               03296
      END;                                                          03297
   !ffffp(pmfjfn); %get a free page%                                03298
   r2 _ r1; %pmap destination%                                      03299
   !POP s,r3; %access%                                              03300
   !POP s,r1; %page handle%                                         03301
   !pmap();                                                         03302
   IF removf AND NOT tops20flag THEN !pmap(-1);                     03303
   !EXCH r1,r2; %r1 has file page handle%                           03304
   IF NOT removf AND tops20flag THEN !pmap();                       03305
      %gets page back into map%                                     03306
   r2 _ r3; %access%                                                03307
   RETURN;                                                          03308
```

```
   END.                                                                 03309
(hmapin)PROC;                                                           01638
   LOCAL count, fbase, hansav;                                          01639
   %get nls pages back%                                                 01640
      FOR count _ (fbase _ $bfree/1000B) UP UNTIL >= $efree/1000B DO
                                                                        01641
         BEGIN                                                          01642
         IF ( hansav _ freemap[count-fbase]) # -1 THEN                  01643
            BEGIN                                                       01644
            r1 _ hansav.fhand;                                          01645
            r3  _ 0;                                                    01646
            r3.hndacc _ hansav.hndacc;                                  01647
            END                                                         01648
         ELSE                                                           01649
            BEGIN                                                       01650
            r1 _ -1;                                                    01651
            r3 _ 120400B6;                                              01652
            END;                                                        01653
         r2.LH _ 4B5;                                                   01654
         r2.RH _ count;                                                 01655
         !JSYS pmap;                                                    01656
         END;                                                           01657
   %clear first cell of freemap as flag that high seg not mapped out%
                                                                        03310
      freemap _ 0;                                                      03311
   RETURN;                                                              01658
   END.                                                                 01659
(prgetst) PROCEDURE;                                                    02060
   % returns byte pointer + count for next statement %                  02061
   LOCAL stid, sdb;                                                     02062
   REF sdb;                                                             02063
   IF (stid _ seqgen(&prseqwk)) = endfil THEN RETURN(endfil);           02064
   IF stid.stastr THEN RETURN(stid.stpsid+1+4407B8, [stid.stpsid].L)
                                                                        02065
   ELSE                                                                 02066
      BEGIN                                                             02067
      IF NOT lodprop( stid, txttyp : &sdb) THEN                         02364
         err($"No text block associated with node");                    02365
      RETURN(&sdb + sdbhdl + 4407B8, sdb.schars);                       02069
      END;                                                              02070
   END.                                                                 02071
                                                                        02072
(oprchr) PROCEDURE;  % get a character for OP %                         01423
   IF opccpos = opcmax THEN RETURN(ENDCHR);                             01424
   BUMP opccpos;                                                        01425
   RETURN(^opcbp);                                                      01426
   END.                                                                 01427
                                                                        01428
(oprnst) PROCEDURE;  % get next statement for OP %                      01429
   LOCAL stid, sdb;                                                     01430
   REF sdb;                                                             01431
   IF (stid _ seqgen(&prseqwk)) # endfil THEN                           01432
      BEGIN                                                             01433
      IF NOT stid.stastr THEN                                           02325
         BEGIN                                                          02326
         IF NOT lodprop( stid, txttyp : &sdb) THEN                      02366
```

```
            err($"No text block associated with node");          02367
         opbp _ opcbp _ &sdb + sdbhdl + 440788;                  02328
         opcmax _ sdb.schars + 1;                                02329
         END                                                     02330
      ELSE                                                       02331
         BEGIN                                                   02332
         opbp _ opcbp _ chbmty + stid.stpsid;;                   02333
         opcmax _ [stid.stpsid].L + 1;                           02334
         END;                                                    02335
      opccpos _ 1;                                               01437
      opnewst _ TRUE;                                            01438
      END;                                                       01439
   RETURN (stid, opbp, opcmax);                                  01440
   END.                                                          01441
                                                                 01442
(oprtxt) %***% PROCEDURE;    % get char pos of 1st text after "name"
%                                                                02349
   LOCAL TEXT POINTER tp;                                        02350
   IF prseqwk.swcstid.stastr THEN RETURN(FALSE);                 02351
   tp _ prseqwk.swcstid;                                         02352
   tp[1] _ fchtxt(tp);                                           02353
   FOR tp[1] DOWN UNTIL <=1 DO                                   02354
      BEGIN                                                      02355
      !IBP opcbp;                                                02356
      BUMP opccpos;                                              02357
      END;                                                       02358
   RETURN(opccpos);                                              02359
   END.                                                          02360
                                                                 02361
(oprgps) PROCEDURE;                                              01456
   RETURN (opccpos);                                             01457
   END.                                                          01458
                                                                 01459
(oprrst) PROCEDURE   % set to passed char position %             01460
   (pos);                                                        01461
   IF pos = -1 THEN opccpos _ opcmax                             01462
   ELSE opccpos _ pos;                                           01463
   opcbp _ chbptr(pos - 1) + opbp.RH -1;                         01466
   RETURN;                                                       01468
   END.                                                          01469
                                                                 01470
(oprlev) PROCEDURE;                                              01471
   oplev _ prseqwk.swclvl;                                       01474
   RETURN ( oplev);                                              01477
   END.                                                          01478
                                                                 01479
(oprsvc) PROCEDURE;                                              01480
   stvect (prseqwk.swcstid, prseqwk.swsvw);                      01481
   RETURN (prseqwk.swsvw);                                       01482
   END.                                                          01483
                                                                 01484
(oprsig) PROCEDURE;                                              01485
   *lit* _ NULL;                                                 01486
   fechsig (prseqwk.swcstid, $lit);                              01487
   RETURN ($lit);                                                01488
   END.                                                          01489
```

```
                                                                    01490
(oprhdf) PROCEDURE;                                                 01491
   RETURN (getfhd (prseqwk.swcstid));                               01492
   END.                                                             01493
                                                                    01494
(oprsid) PROCEDURE;                                                 03269
   RETURN (getsid (prseqwk.swcstid));                               03270
   END.                                                             03271
                                                                    03272
(oprdtc) PROCEDURE; % put date and time of creation into string %
                                                                    02957
   LOCAL sdbadr;                                                    03231
   REF sdbadr;                                                      03232
   lit.L _ 0;                                                       03233
   IF NOT prseqwk.swcstid.stastr THEN                               03234
      BEGIN                                                         03235
      IF NOT lodprop( prseqwk.swcstid, txttyp :&sdbadr) THEN        03273
         err($" No text block with this node");                     03274
      dtfrmt( sdbadr.stime, $lit );                                 03237
      END;                                                          03238
   RETURN ($lit);                                                   02958
   END.                                                             02959
                                                                    02960
(oprdc) PROCEDURE; % put date of creation into string %             03215
   LOCAL sdbadr;                                                    03239
   REF sdbadr;                                                      03240
   lit.L _ 0;                                                       03241
   IF NOT prseqwk.swcstid.stastr THEN                               03242
      BEGIN                                                         03243
      IF NOT lodprop( prseqwk.swcstid, txttyp :&sdbadr) THEN        03275
         err($" No text block with this node");                     03276
      daofrmt( sdbadr.stime, $lit );                                03245
      END;                                                          03246
   RETURN ($lit);                                                   03247
   END.                                                             03248
                                                                    03249
(oprcid) PROCEDURE; % put ident of last editor into string %        03219
   lit.L _ 0;                                                       03256
   IF NOT prseqwk.swcstid.stastr THEN                               03257
      BEGIN                                                         03258
      idfrmt ( prseqwk.swcstid, $lit );                             03265
      END;                                                          03261
   RETURN ($lit);                                                   03262
   END.                                                             03263
                                                                    03264
(oprint) PROCEDURE; % return internal id format for statement %     03223
   LOCAL sdbadr, stdb;                                              03280
   REF sdbadr;                                                      03281
   IF NOT lodprop( prseqwk.swcstid, txttyp :&sdbadr, stdb) THEN     03278
      err($" No text block with this node");                        03279
   RETURN (getint (stdb));                                          03224
   END.                                                             03225
                                                                    03226
(oprdt) PROCEDURE; % return internal date and time format for
statement %                                                         03227
   LOCAL sdbadr;                                                    03266
```

```
    REF sdbadr;                                                          03267
    lodprop( prseqwk.swcstid, txttyp :&sdbadr);                          03277
    RETURN (sdbadr.stime);                                               03228
    END.                                                                 03229
                                                                         03230
(opinit) PROCEDURE (da, jfn, devtyp, opflags, adgcout, comdev);          03322
    oprwrk _ jfn;                                                        03323
    oprwrk[1] _ devtyp;                                                  03324
    oprwrk[2] _ da;                                                      03325
    oprwrk[3] _ $oprchr;                                                 03326
    oprwrk[4] _ $oprtxt;                                                 03327
    oprwrk[5] _ $oprrst;                                                 03328
    oprwrk[6] _ $oprgps;                                                 03329
    oprwrk[7] _ $oprlev;                                                 03330
    oprwrk[8] _ $oprsvc;                                                 03331
    oprwrk[9] _ $oprsig;                                                 03332
    oprwrk[10] _ $oprhdf;                                                03333
    oprwrk[11] _ $oprnst;                                                03334
    oprwrk[12] _ opflags;                                                03335
    oprwrk[13] _ adgcout;                                                03336
    oprwrk[14] _ $oprsid;                                                03337
    oprwrk[15] _ comdev;                                                 03338
    oprwrk[16] _ $oprdtc;                                                03339
    oprwrk[17] _ $oprdc;                                                 03340
    oprwrk[18] _ $oprcid;                                                03341
    oprwrk[19] _ $oprint;                                                03342
    oprwrk[20] _ $oprdt;                                                 03343
    RETURN;                                                              03344
    END.                                                                 03345
                                                                         03346
(getsbn) PROCEDURE (astrng);                                             01512
    %convert string passed to sixbit name%                               01513
    LOCAL sixbit, charct, bytptr;                                        01514
    REF astrng;                                                          01515
    bytptr _ 688 + $sixbit - 1;                                          01516
    sixbit _ 0;                                                          01517
    charct _ empty;                                                      01518
    WHILE ((charct _ charct + 1) <= 6) AND (charct <= astrng.L) DO
                                                                         01519
      ^bytptr _ (*astrng*[charct] + 40B) .A 77B;                         01520
    RETURN(sixbit);                                                      01521
    END.
                                                                         01522
FINISH of SEQFIL                                                         01523
```

SEQGEN

```
< NLS, SEQGEN.NLS;4, >, 7-MAR-77 17:50 DSM ;;;;
FILE seqgen  %  L10 <rel-nls>seqgen  %% (L10,) (rel-nls,seqgen.rel,) %
                                                                      02
                                                                    0807
% Routines making up the sequence generator %                         04
                                                                    0749
   % OPENSEQ is used to open a sequence
     CLOSESEQ is used to close a sequence
     SEQGEN is used to get the next statement in a sequence
     There is also a routine SEQNXT which gets the next statement in a
   sequence considering all the viewspecs except the content analyzer
   ones %                                                           0750
                                                                    0753
   SET sqsksz = 2000B;  % seqgen stack size %                       01145
   REGISTER  s=9, m=10;                                             0657
   REF sqsavsw;   % global for saving argument (address of sequence work
   area) while switching stacks %                                   01086
                                                                    01087
   REF tda; % global pointing to display area for tty %             01297
% the Sequence Generator %                                           018
                                                                    0698
   (openseq) PROCEDURE    % open a sequence %                       0697
      (fstid,   % STID with which to begin the sequence %           0700
      lbstid,   % STID of statement heading last branch in
                  the sequence%                                     0701
      vspec,    % first word of viewspecs to use %                  0702
      vspc2,    % 2nd word of viewspecs %                           0703
      usqcod,   % address of user seqgen routine -- or 0 %          0704
      cacode);  % address of CONAN routine  -- or 0 %               0705
                                                                    0706
      % OPENSEQ                                                    0699
         Gets a sequence work area (with its attendent stack and
         statement vector work area) and returns the address of that
         work area.                                                 0713
         Initializes the work area                                  0715
         If generating statement numbers, initializes the statement
         vector work area also.                                     0722
         Perhaps calls the user's seqgen (with the address of the work
         area and telling it that its being called as openseq)      0714
         The stack is not switched which means that user openseq is not
         allowed to do port-sends. %                                0707
                                                                    0708
      LOCAL sw;                                                     0710
      REF sw;                                                       0711
                                                                    0712
      % allocate a work area, a stack, and a statement vector work area%
                                                                    0754
         &sw _ getsgw();                                            0709
      % initialize the work area %                                  0716
         IF fstid.stastr  THEN   % stid points to a string %        01094
            BEGIN                                                   01095
            sw.swstid  _ sw.swcstid _ fstid;                        01098
            sw.swclvl  _ 0;                                         01110
            sw.swvspec _ vspec;                                     01114
            sw.swvsp2  _ vspc2;                                     01116
            END                                                     01099
```

```
      ELSE                                                          0925
         BEGIN                                                      01108
         IF getsid (fstid) = 0 THEN   % illegal stid %              01101
            BEGIN                                                   01102
            sw.swcstid.stfile _ fstid.stfile;                       01103
            sw.swcstid.stpsid _ origin;                             01104
            sw.swstid _ endfil;                                     01105
            END                                                     01106
         ELSE sw.swstid _ sw.swcstid _ fstid;                       01107
         % set up level and statement number stuff %                 0733
         sw.swclvl _                                                 0763
            IF vspec.vsstnf THEN stvect (sw.swcstid, sw.swsvw)       0734
            ELSE getlev (sw.swcstid);                                0736
         % set up viewspecs -- maybe relative level stuff %          0810
            sw.swvspec  _                                            0719
               IF vspec.vsrlev THEN reslev (vspec, sw.swclvl)        0738
               ELSE vspec;                                           01173
            sw.swvsp2  _ vspc2;                                      0720
         END;                                                        01109
      sw.swlbstid _ lbstid;                                          0718
      sw.swusqcod _ usqcod;                                          0728
      sw.swcacode _ cacode;                                          0729
      sw.swslvl   _ sw.swclvl;   % remember what level started at %
                                                                     01117
      sw.swkflg   _ FALSE;   % nothing has been sent or passed yet %
                                                                     0931
      % start seqgen coroutine first time at its initialization port,
      i.e., CALL it %                                                0769
         sw.swcall _ TRUE;                                           0768
   % maybe call the users seqgen code as openseq %                  0770
      IF vspec.vsusqf AND sw.swusqcod THEN                           0771
         [sw.swusqcod] (&sw, sqopn);                                 01121
   RETURN (&sw);                                                     0746
   END.                                                              0747
                                                                     0748
(closeseq) PROCEDURE   % close a sequence %                          0773
   (sw);   % address of the sequence work area to close %           0774
                                                                     0775
   % CLOSESEQ                                                       0776
      Check for a legal sequence work area.                          01001
      Perhaps calls the user's seqgen (with the address of the work
      area and telling it that its being called as a closeseq)       0778
      Releases the sequence work area (with its attendent stack and
      statement vector work area)                                    0777
      The stack is not switched which means the user closeseq is not
      allowed to do port-sends %                                     0779
   REF sw;                                                           0781
   % maybe call the users seqgen code as closeseq %                  0794
      IF sw.swvspec.vsusqf AND sw.swusqcod THEN                      0795
         [sw.swusqcod] (&sw, sqcls);                                 01122
   % release the work area, stack, and statement vector work area %
                                                                     0792
      relsgw (&sw);                                                  0793
   RETURN;                                                           0804
   END.                                                              0805
                                                                     0806
```

```
(seqgen) PROCEDURE                                                    0848
   (sw);   % address of a sequence generator work area %              0864
                                                                      0849
   % SEQGEN                                                           0850
      Calling SEQGEN results in the caller being port-sent (looks
      like a normal return to the caller) the next "STID" in that
      sequence -- actually it may be an ENDFIL (no more items in this
      sequence), or a stastr, or the STID of a statement.             0934
      Also the following fields of the work area are updated:         0942
         SWCSTID (remains unchanged if an ENDFIL or stastr is the
         item returned)                                               0949
         SWSTID (will be same as thing returned)                      0943
         SWSRSAV and SWMRSAV; SWCLVL; SWCALL, SWKFLG                  0945
         (unless conan code is using sport rather than using send or
         returning with flag set).                                    0950
      It checks for a legal sequence work area.                      01002
      It also updates the statement vector if necessary.              0974
      SEQGEN is actually merely a dispatch routine -- it calls either
      usqcod (user sequence code) or SSEQGEN (system sequence
      generator);                                                     0863
      It effects one-half of the coroutine linkage involved in
      port-sends (SPORT effects the other half).                      0851
      The input fork is informed that the Sequence Generator will
      handle rubouts.  Since the only way out of here is thru SPORT,
      the occurance of a rubout is checked for there and an ENDFIL
      sent if one occured.                                            0932
      The stacks are switched here.  Thus no locals are allowed and
      the argument must be saved in a global. %                       0852
                                                                      0853
   REF sw;                                                            0927
   IF sw.swstid = endfil THEN RETURN (endfil);                       01008
   rubabt _ FALSE;  % tells input fork not to abort if see rubout %
                                                                      0951
   &sqsavsw _ &sw;       % save the argument in a global %           0855
   % switch stacks %                                                 0856
      s _ sqsavsw.swsrsav := s;                                       0857
      m _ sqsavsw.swmrsav := m;                                       0858
   % depending on swcall (TRUE if last item in sequence was recieved
   by a normal return, FALSE it was recieved via a port-send), the
   next item in sequence is asked for by a CALL or a RETURN.  Note
   that OPENSEQ sets swcall TRUE and this is the only other place in
   NLS where it is tested or set. %                                   0859
      IF sqsavsw.swcall := FALSE THEN                                 0865
         BEGIN                                                        0866
         % call a user seqgen or SSEQGEN %                           0875
         IF sqsavsw.swusqcod AND sqsavsw.swvspec.vsusqf THEN          0867
            [sqsavsw.swusqcod] (&sqsavsw, sqgnxt)                    01085
         ELSE sseqgen (&sqsavsw);                                     0873
         sqsavsw.swcall _ TRUE;    % only get here if normal return
         made (not a port-send) %                                     0868
         sport (&sqsavsw);    % SPORT will switch the stacks back and
         return to SEQGEN's caller %                                  0869
         END                                                          0870
      ELSE RETURN;   % thru port, not to SEQGEN's caller %;           0871
   END.                                                               0861
                                                                      0862
```

```
(sseqgen) PROCEDURE    % NOBODY BUT SEQGEN SHOULD CALL THIS ROUTINE %
                                                                    0571
   (sw);   % address of a sequence generator work area %            0876
                                                                    0882
   % SSEQGEN                                                        0877
      Given address of sequence work area this procedure returns the
      next item in that sequence.                                    021
      SSEQGEN or SEQNXT or SEND or ucacod stores that value in SWSTID
      (and SWCSTID if it is not ENDFIL or a stastr).                0952
      The routine SEQNXT finds the next STID, considering all
      viewspecs except content anlysis ones -- SSEQGEN takes care of
      those.                                                         035
      Note that SEQNXT is not called the first time thru this
      routine.  Thus the first stid returned will be the one the
      sequence was initialized with -- unless conan code fails it.
                                                                    0957
      Rubouts are checked for after every call on conan code. %      042
                                                                    0884
   LOCAL stid;                                                      0189
   REF sw;                                                          0574
   LOOP                                                              064
      BEGIN                                                         0886
      WHILE (sw.swvspec.vscapf OR (sw.swvspec.vscakf AND NOT
      sw.swkflg)) AND sw.swcacode > 0 AND sw.swstid # endfil DO      0192
         % (conan must pass statements viewspec is on OR (turn off
         conan after first passed statement viewspec is on BUT
         haven't passed one yet)) AND there is a conan program AND we
         haven't reached the end of the sequence yet %              0975
         BEGIN                                                      0954
         stid _ sw.swcstid;                                         0194
         FIND SF(stid);                                             0198
         IF [sw.swcacode] (&sw) OR inptrf %rubout% THEN             0204
            BEGIN                                                  01147
            sw.swstid _ stid;  % a send from this same statement will
            have blown this field %                                01149
            sport (&sw);                                           01146
            END;                                                   01148
         seqnxt (&sw);                                              0959
         END;                                                       0960
      sport (&sw);                                                  0964
      seqnxt (&sw);                                                 0885
      END;                                                           063
   END.                                                              066
                                                                     067
(seqnxt) PROCEDURE                                                  0127
   (sw);   % address of sequence work area %                        0980
                                                                    0981
   % SEQNXT                                                         0982
      Returns the next STID in this sequence (or ENDFIL if no more).
                                                                    0128
      It updates the following fields of the work area:             0811
         swstid, swcstid (if not ENDFIL), and swclvl.               0812
      It also updates the statement vector if necessary.            0813
      It takes into account all viewspecs, except content analysis,
      during the search. %                                          0131
                                                                    0983
```

```
   LOCAL stid, %current stid%                                          0135
      vspec;   %first viewspec word (sw.swvspec)%                      0136
   REF sw;                                                             0137
   IF sw.swcstid.stastr THEN RETURN (sw.swstid _ endfil);             01100
   vspec _ sw.swvspec;                                                 0139
   IF sw.swclvl >= vspec.vslev                                         0140
      OR (stid _ getsub (sw.swcstid)) = sw.swcstid THEN               01012
      BEGIN %see if superstructure fits viewspecs%                     0142
      IF (stid _ sw.swcstid) = sw.swlbstid THEN                        0143
         RETURN (sw.swstid _ endfil);                                  0984
      WHILE getftl (stid) DO                                           0145
         BEGIN                                                         0146
         stid _ getsuc (stid);                                         0147
         BUMP DOWN sw.swclvl;                                          0150
         IF stid = sw.swlbstid THEN                                   01014
            RETURN (sw.swstid _ endfil);                              01016
         IF vspec.vsstnf THEN stvmod (sw.swsvw, up);                   0151
         END;                                                          0152
      IF stid.stpsid = origin THEN RETURN (sw.swstid _ endfil);       01120
      %successor is next PSID in sequence%                             0156
      stid _ getsuc (stid);                                            0157
      IF vspec.vsstnf THEN stvmod (sw.swsvw, suc);                     0158
      END                                                              0159
   ELSE                                                                0160
      BEGIN   %substructure fits viewspecs%                            0161
      BUMP sw.swclvl;   %increase level%                               0162
      IF vspec.vsstnf THEN stvmod (sw.swsvw, sub);                     0163
      END;                                                             0164
   RETURN (sw.swcstid _ sw.swstid _ stid);                             0170
   END.                                                                0171
                                                                       0173
(seqend) PROCEDURE (stid, vs1, vs2);                                  01316
   % This procedure may be used to find the last branch to be handled
   by the sequence generator depending on the branch or plex only
   viewspecs;  the sequence generator itself no longer checks the
   veiwspecs.  Thus, the stid of the last branch must be passed as a
   parameter to openseq! %                                            01317
   LOCAL retstd;                                                      01318
   IF stid.stastr THEN RETURN(endfil);                                01319
   retstd _ CASE TRUE OF                                              01320
      = vs1.vsbrof: stid;                                             01321
      = vs1.vsplxf : getail(stid);                                    01322
      ENDCASE endfil;                                                 01323
   RETURN(retstd);                                                    01324
   END.
                                                                      01325
(send) PROCEDURE                                                       0634
   (sw,    % address of a sequence work area %                         0846
   str);   % an ENDFIL or address of a string %                        0847
                                                                       0840
   % SEND                                                             0841
      Takes the place of the old send.                                 0842
      Sets the swstid (but not the swcstid) field of the work area.
                                                                       0843
      Calls sport. %                                                   0844
                                                                       0845
```

```
   REF sw, str;                                                    0636
   IF &str = endfil THEN sw.swstid _ endfil                        0816
   ELSE                                                           01018
      BEGIN                                                       01019
      sw.swstid.stastr _ TRUE;                                     0669
      sw.swstid.stadr  _ &str;                                     0817
      IF str.L = empty THEN sw.swstid.stadr _ $" "; %no null strings
      permitted%                                                   0670
      END;                                                        01020
   sport (&sw);                                                    0818
   RETURN;                                                        01021
   END.                                                            0648
                                                                   0827
(sport)  PROCEDURE    % send-port mechanism %                       0819
   (sw);    % address of a sequence work area %                  01022
                                                                  01093
   % SPORT                                                        0835
      Effects one-half of the coroutine linkage involved in
      port-sends (SEQGEN effects the other half).                  0836
      Control o's and rubouts are checked for here.               01138
      The stacks are switched here. %                              0837
                                                                   0838
   REF sw;                                                        01083
   sw.swkflg _ TRUE; % something has been returned in this sequence %
                                                                   0930
   % the stacks are about to be switched so save everything needed
   later in globals %                                              0831
      &sqsavsw _ &sw;                                              0829
   % switch stacks %                                              0833
      s _ sqsavsw.swsrsav := s;                                    0638
      m _ sqsavsw.swmrsav := m;                                    0639
   IF inptrf THEN   % a contol o or a rubout has occurred %       01139
      BEGIN                                                       01141
      %rubabt _ TRUE;% % Why was this here??? %                   01143
      sqsavsw.swstid _ endfil;                                    01140
      END;                                                        01144
   % the following RETURN has the effect of returning from the
   procedure last called from the now-in-effect stack %            0834
   RETURN (sqsavsw.swstid);                                        0830
   END.                                                            0826
                                                                   0828
(sqinit) PROCEDURE;   % sequence generator init %                 01355
                                                                  01356
   % SQINIT                                                      01357
      Is only called at INIT time.                                01358
      It initializes all the sequence work areas (including the
      attached stacks and sequence vector work areas). %          01359
                                                                  01360
   LOCAL sw, swcnt;    % address of a sequence generator work area %
                                                                  01361
   REF sw;                                                        01362
   swcnt _ 0;                                                     01363
   FOR &sw _ $sqgwas UP $sqwrkl UNTIL >= $sqgaend DO              01364
      BEGIN                                                       01365
      sw.swalloc _ FALSE;   % allocation bit %                    01366
      IF swcnt < $sqstkn THEN                                     01367
```

```
         BEGIN                                                          01368
         sw.swstkdec _ TRUE;                                            01369
         sw.swstkalloc _ FALSE;                                         01370
         sw.swsvw _ $sqsvws + (swcnt * (svmxlev + 1));                  01371
         sw.swstkloc _ $sqstks + (swcnt * $sqsksz);                     01372
         [sw.swstkloc] _ $uflow;                                        01373
         sw.swstksize _ $sqsksz;                                        01374
         END                                                            01375
      ELSE                                                              01376
         BEGIN                                                          01377
         sw.swstkdec _ FALSE;                                           01378
         sw.swstkalloc _ FALSE;                                         01379
         sw.swsvw _ sw.swstkloc _ sw.swstksize _ 0;                     01380
         END;                                                           01381
      BUMP swcnt;                                                       01382
      END;                                                              01383
   RETURN;                                                              01384
   END.
                                                                        01385
(getsgw) PROCEDURE;   % get a sequence generator work area %           01386
                                                                        01387
   % GETSGW                                                            01388
      Allocates, initializes some parts of, and returns the address
      of a sequence generator work area (a sequence work area, a
      statement vector work area, and a stack). %                       01389
                                                                        01390
   LOCAL sw, swcnt, blkaddr, svwaddr;    % address of a sequence
   generator work area %                                                01391
   REF sw;                                                              01392
   swcnt _ 0;                                                           01393
   FOR &sw _ $sqgwas UP $sqwrkl UNTIL >= $sqgaend DO                    01394
      IF NOT sw.swalloc THEN                                            01395
         BEGIN                                                          01396
         sw.swalloc _ TRUE;   % allocation bit %                        01397
         IF sw.swstkdec THEN                                            01398
            BEGIN                                                       01399
            sw.swsrsav.RH _ sw.swmrsav.RH _ $sqstks + (swcnt *
            $sqsksz);                                                   01400
            sw.swsrsav.LH _ sw.swmrsav.LH _ -$sqsksz;                   01401
            END                                                         01402
         ELSE                                                           01403
            BEGIN                                                       01404
            IF NOT (blkaddr _ getarray($sqsksz-bhl-1, $dspblk)) THEN
                                                                        01405
               SIGNAL (sqerr, $"no more sequence work areas");          01406
            sw.swstkalloc _ TRUE;                                       01407
            sw.swstkloc _ blkaddr;                                      01454
            sw.swsrsav.RH _ sw.swmrsav.RH _ blkaddr;                    01408
            sw.swsrsav.LH _ sw.swmrsav.LH _ -($sqsksz-bhl-1);           01409
            [blkaddr] _ $uflow;                                         01410
            IF NOT (svwaddr _ getarray(svmxlev+1, $dspblk)) THEN
                                                                        01411
               SIGNAL (sqerr, $"no more sequence work areas");          01412
            sw.swsvw _ svwaddr;                                         01413
            END;                                                        01414
         RETURN (&sw);                                                  01415
```

```
        END                                                          01416
      ELSE BUMP swcnt;                                               01417
   SIGNAL (sqerr, $"no more sequence work areas");                   01418
   END.
                                                                     01419
(relsgw) PROCEDURE    % release a sequence generator work area %
                                                                     01433
   (sw);     % address of a sequence generator work area %          01434
                                                                     01435
   % RELSGW                                                         01436
      Deallocates a sequence generator work area (a sequence work
      area, a statement vector work area, and a stack). %            01437
                                                                     01438
   REF sw;                                                           01439
   IF NOT (&sw IN [$sqgwas, $sqgaend]                                01440
      AND ((&sw - $sqgwas) MOD $sqwrkl) = 0 AND sw.swalloc) THEN
                                                                     01441
         SIGNAL (sqerr, $"illegal sequence work area");              01442
   IF sw.swstkalloc THEN                                             01443
      BEGIN                                                          01444
      IF NOT freeblk((sw.swstkloc-bhl),$dspblk) THEN                 01445
         SIGNAL (sqerr, $"unable to free sequence work area stack");
                                                                     01446
      IF NOT freeblk((sw.swsvw-bhl),$dspblk) THEN                    01447
         SIGNAL (sqerr, $"unable to free sequence work area statement
         vector");                                                   01448
      sw.swstkalloc _ FALSE;                                         01453
      sw.swstkloc _ sw.swsvw _ 0;                                    01455
      END;                                                           01452
   sw.swalloc _ FALSE;                                               01449
   RETURN;                                                           01450
   END.
                                                                     01451
% SEQGEN, STATEMENT NUMBER / VECTOR / SIGNATURE UTILITY ROUTINES %   0263
   (cpysw) PROCEDURE  % copy parts of one sequence work area to another%
                                                                     01124
      (fromsw, tosw);   % record pointers to work areas %           01125
         % this has the effect of making the tosw point to the same item
         as the fromsw %                                             01126
      REF fromsw, tosw;                                              01127
                                                                     01128
      tosw.swcstid _ fromsw.swcstid;                                 01129
      tosw.swstid  _ fromsw.swstid;                                  01130
      tosw.swclvl  _ fromsw.swclvl;                                  01134
      IF tosw.swvspec.vsstnf THEN mvbfbf (fromsw.swsvw, tosw.swsvw,
      svmxlev);                                                      01174
      RETURN;                                                        01135
      END.                                                           01136
                                                                     01137
   (pjseqg) PROCEDURE (sw, which); % Print Journal Seq Generator %   01175
      LOCAL pjvs1, pjvs2, pjopnusqc, sv1, sv2, adstr[40], pjmstid;   01177
      LOCAL TEXT POINTER pjtp;                                       01179
      LOCAL STRING pjstn [200];                                      01178
      REF sw, pjsw, pjlsw;                                           01176
      CASE which OF                                                  01180
         =sqopn: % called at open %                                  01181
```

```
        BEGIN                                                   01182
        pjrbab _ rubabt; % save current RUBOUT disposition %    01183
        rubabt _ FALSE; % and disable RUBOUT %                  01184
        pjrubout _ FALSE; % init RUBOUT indicator %             01185
        pjopnusqc _ IF pjsavf THEN pjusqc ELSE 0; % assure user seq
        gen control (if any ) if appropriate viewspec on %      01186
        &pjsw _ openseq (sw.swcstid, sw.swlbstid, sw.swvspec,
        sw.swvsp2, pjopnusqc, sw.swcacode); % open secondary
        sequence %                                              01187
        pjstid _ FALSE; % record not in linked-to file %        01188
        RETURN; % return to caller %                            01189
        END;                                                    01190
      =sqgnxt: NULL; % called for next in seq -- fall through % 01191
      =sqcls: % called at close %                               01192
        BEGIN                                                   01193
        closeseq (&pjsw); % close secondary sequence %          01194
        IF pjstid THEN % if in a linked-to file %               01195
          BEGIN                                                 01196
          closeseq (&pjlsw); % close that sequence %            01197
          close (pjstid.stfile); % and it's file %              01198
          END;                                                  01199
        rubabt _ pjrbab; % restore RUBOUT disposition %         01200
        RETURN; % and return to caller %                        01201
        END;                                                    01202
      ENDCASE err (); % called for any other purpose -- error % 01203
    LOOP % here for each statement in primary file %            01204
      BEGIN                                                     01205
      IF pjrubout THEN sw.swstid _ endfil % if RUBOUT hit, force
      endfil %                                                  01206
      ELSE                                                      01207
        BEGIN                                                   01208
        seqgen (&pjsw); % fetch next in primary file %          01209
        cpysw (&pjsw, &sw); % copy for caller %                 01210
        END;                                                    01211
      sport (&sw); % return it to him %                         01212
      pjtp _ pjsw.swstid; % build text pointer %                01213
      pjtp[1] _ 1; % to start of statement %                    01214
      IF FIND SF(pjtp) ["Location: "] ^pjtp '( ['')] THEN       01215
        BEGIN % if this statement for a Journal FILE %          01216
        IF (pjmstid_getsub(pjtp)) # pjtp THEN                   01326
          BEGIN %if message already there, don't go after link%
                                                                01327
          pjmstid _ getail(pjmstid);                            01328
          IF FIND SF(pjmstid) ("Message:") THEN REPEAT LOOP;    01329
          END;                                                  01330
        ON SIGNAL                                               01217
          =errsig: % if any error is encountered %              01218
            BEGIN                                               01219
            pjsep (&sw, $"Document has been Archived"); % Assume
            the file is not on-line %                           01220
            REPEAT LOOP; % and abandon it %                     01221
            END;                                                01222
          ELSE;                                                 01223
        lnkpspc (1, $pjtp, $stno, $stn2, $pjstn, $num, $adstr); %
        parse the link %                                        01224
        pjstid _ nfstid ($stn2, $pjstn, $num); % open the secondary
```

```
        file %                                                          01225
        ON SIGNAL ELSE;                                                 01226
        pjvs1 _ tda.davspec; % save the display %                       01227
        pjvs2 _ tda.davspc2; % area viewspecs %                         01228
        tda.davspec.vsbrof _ FALSE;                                     01229
        feedlt (&tda, $num); % build viewspecs from link %              01230
        pjopnusqc _ IF pjsavf THEN pjusqc ELSE 0; % assure user seq
        gen control (if any ) if appropriate viewspec on %              01231
        &pjlsw _ openseq (pjstid, seqend(pjstid, sv1 _ tda.davspec,
        sv2 _ tda.davspc2), sv1, sv2, pjopnusqc, sw.swcacode); %
        open tertiary sequence using them %                             01232
        tda.davspec _ pjvs1; % restore the display %                    01233
        tda.davspc2 _ pjvs2; % area viewspecs %                         01234
        IF NOT pjrubout THEN                                            01235
           BEGIN                                                        01236
           pjsep (&sw, $"Text of Cited Document Follows"); % output
           document header %                                            01237
           LOOP % here for each cited document %                        01238
              BEGIN                                                     01239
              IF pjrubout THEN % if RUBOUT hit %                        01240
                 BEGIN                                                  01241
                 pjrubout _ FALSE; % reset RUBOUT indicator %           01242
                 send (&sw, $"..."); % show that the document was
                 aborted %                                              01243
                 EXIT LOOP; % and be finished with it %                 01244
                 END;                                                   01245
              seqgen (&pjlsw); % fetch next of cited document %
                                                                        01246
              IF pjlsw.swstid = endfil THEN % if end of document %
                                                                        01247
                 BEGIN                                                  01248
                 rubabt _ FALSE; % we know that 'sport' reset
                 'rubabt' %                                             01249
                 EXIT LOOP; % end of tertiary sequence %                01250
                 END;                                                   01251
              cpysw (&pjlsw, &sw); % if no RUBOUT, return next %
                                                                        01252
              sport (&sw); % of tertiary sequence to caller %           01253
              END;                                                      01254
           pjsep (&sw, $"End of Cited Document"); % output document
           trailer %                                                    01255
           END;                                                         01256
        closeseq (&pjlsw); % close tertiary sequence %                  01257
        close (pjstid.stfile); % and its file %                         01258
        pjstid _ FALSE; % mark no tertiary seq in progress %            01259
        END;                                                            01260
     END;                                                               01261
  END.                                                                  01262
                                                                        01263
(pjsep) PROCEDURE (sw, title); % Print Journal Header/Trailer %         01264
   REF sw, title;                                                       01265
   LOCAL len, i, j, hinc, end;                                          01266
   LOCAL STRING str [300];                                              01267
   % get horizontal increment and line length from display area %
                                                                        01268
      hinc _ tda.dahinc; % horizontal increment %                       01269
```

```
      len _ tda.damcol-hinc; % length of print line %                    01270
   % force the sequence generator to make this line a level 1 thing %
                                                                         01271
      sw.swclvl _ 1; % force statement level to highest %                01272
   % edit the separator string %                                        01273
      % start with a blank line %                                       01274
         *str* _ EOL;                                                   01275
      % left of title %                                                 01280
         end _ (j_len-(2+title.L)*hinc)/2;                              01313
         FOR i _ hinc UP hinc UNTIL >= end DO                           01281
            *str* _ *str*, '<;                                          01309
      % title %                                                         01282
         *str* _ *str*, SP, *title*, SP;                                01283
      % right of title %                                                01284
         end _ len -2*hinc;                                             01314
         FOR i _ i+title.L*hinc + 2*hinc UP hinc UNTIL >= end DO
                                                                        01285
            *str* _ *str*, '>;                                          01310
      % lower left corner %                                            01286
         *str* _ *str*, EOL;                                            01287
   % send the string to the seq. gen. %                                 01292
      send (&sw, $str); % return header/trailer to caller %             01293
   RETURN;                                                              01294
   END.                                                                 01295
                                                                        01296
(reslev) PROCEDURE(vspec, clevel);                                       0672
   %Given a viewspec word and a level, this routine will perform any
   relative level adjustment required by the viewspec word and return
   the updated viewspec word.%                                          0673
   IF vspec.vsrlev THEN                                                 0674
      BEGIN                                                             0106
      IF vspec.vslevd THEN % up %                                       0107
         vspec.vslev _ MAX (clevel-vspec.vslev, 1)                      0108
      ELSE % down %                                                     0109
         vspec.vslev _ MIN (clevel+vspec.vslev, 63);                    0110
      vspec.vsrlev _ FALSE;                                             0111
      vspec.vslevd _ FALSE;                                             0112
      END;                                                              0115
   RETURN(vspec);                                                       0675
   END.

                                                                        0677
(stvect)                                                                0265
   %This routine generates a statement vector. It is called with a
   STID as the first argument and the address of the first word of a
   64 word work area as the second. The number of words in the work
   area determine how many levels down in structure the routine can
   go (currently 64). ERR(5) is called if the work area is not long
   enough to contain the vector for the given STID. STVECT uses
   routine STPOS.STVECT returns an integer, which is the level of the
   statement. It also creates a statement vector. Upon completion,
   the first word of the work area contains the level of the STID and
   subsequent cells contain the position within the respective
   plexes. Thus for statement 1D2, the vector would contain (3, 1,
   4, 2) in the first four cells.%                                      0266
   %----------------%                                                  0282
```

```
   PROCEDURE (stid, stvwrk);                                          0283
   LOCAL                                                              0284
      curwrd, %current word being built in vector%                    0285
      level;  %level of STID%                                         0286
   REF stvwrk, curwrd;                                                0287
   &curWrd _ &stvwrk + svmxlev;                                       0288
   level _ 0;                                                         0289
   UNTIL stid.stpsid = origin DO %count up to origin%                 0290
      BEGIN                                                           0291
      IF (&curwrd _ &curwrd-1) <= &stvwrk THEN err(5);                0292
      curwrd _ stpos(stid);                                           0293
      stid _ getup(stid);                                             0294
      BUMP level;                                                     0295
      END;                                                            0296
   stvwrk _ level;                                                    0297
   %move vector to top of work area%                                  0298
   mvbfbf (&curwrd, &stvwrk+1, level);                                0299
   RETURN (level);                                                    0300
   END.                                                               0301
                                                                      0302
                                                                      0303
(stvmod)                                                              0304
   %This routine modifies a statement vector.  It is called with the
   address of a statement vector as the first argument, and a type
   parameter as the second. The statement vector is assumed to be
   initialized, and is modified as specified by TYPE.%                0305
   %-----------------%                                               0310
   PROCEDURE (stvwrk, type);                                          0311
   LOCAL word; %current word in statement vector%                     0312
   REF stvwrk, word;                                                  0313
   IF stvwrk = 0 THEN    % treat the origin statement specially %
                                                                      01150
      CASE type OF                                                    01160
         =sub:                                                        01161
            BEGIN                                                     01162
            stvwrk _ 1;                                               01171
            [&stvwrk + 1] _ 1;                                        01172
            END;                                                      01166
         =suc, =pred, =up: NULL;                                      01168
         ENDCASE err(0)                                               01170
   ELSE                                                               01155
      BEGIN                                                           01156
      &word _ &stvwrk + stvwrk;                                        0314
      CASE type OF                                                     0315
         =sub:                                                         0316
            BEGIN                                                      0317
            IF (stvwrk _ stvwrk + 1) > svmxlev THEN err(5);            0318
            [&word + 1] _ 1;                                           0320
            END;                                                       0321
         =suc:  word _ word + 1;                                       0322
         =pred: word _ MAX (word-1, 1);                                0323
         =up:   stvwrk _ MAX (stvwrk-1, 0);                            0324
         ENDCASE err(0);                                               0325
      END;                                                            01159
   RETURN;                                                            0326
   END.                                                               0327
```

```
                                                                    0328
(stpos)                                                             0329
   %Given a STID, this routine returns an integer which is the
   position of that statement within its plex.%                     0330
   %-----------------%                                              0332
   PROCEDURE (stid);                                                0333
   LOCAL                                                            0334
      nstid,                                                        0335
      posit; %current position%                                     0336
   IF getfhd(stid) THEN RETURN (1);                                 0337
   posit _ 1;                                                       0338
   nstid _ gethed(stid);                                            0339
   UNTIL nstid = stid DO                                            0340
      BEGIN                                                         0341
      nstid _ getsuc(nstid);                                        0342
      BUMP posit;                                                   0343
      END;                                                          0344
   RETURN (posit);                                                  0345
   END.                                                             0346
                                                                    0347
                                                                    0348
(getlev)                                                            0349
   %Called with STID, returns level of that statement.%             0350
   %-----------------%                                              0351
   PROC(stid);                                                      0352
   LOCAL level; %current level%                                     0353
   level _ 0;                                                       0354
   UNTIL stid.stpsid = origin OR getorf( stid ) DO                  0355
      BEGIN                                                         0356
      stid _ getup(stid);                                           0357
      BUMP level;                                                   0358
      END;                                                          0359
   RETURN (level);                                                  0360
   END.                                                             0361
                                                                    0362
                                                                    0363
(fechno)                                                            0364
   %Appends statement number of stid to string.  Give the STID as the
   first argument, and the address of the string which is to contain
   the statement number as the second.  The statement number will be
   built in the string.  If the structure is not intact or the
   statement vector cannot be built, a call to RERROR or an EXCEED
   CAPICITY ERROR may result.%                                      0365
   %-----------------%                                              0372
   PROCEDURE(stid, astr);                                           0373
   LOCAL fchsvw[100];                                              01080
   stvect(stid, $fchsvw);                                           0374
   fechnm($fchsvw, astr);                                           0375
   RETURN;                                                          0376
   END.                                                             0377
                                                                    0378
(fechnp)                                                            0379
   %This is like FECHNO, but generates the statement number of the
   successor (whether it exists or not).%                           0380
   %-----------------%                                              0382
   PROCEDURE(stid, astr);                                           0383
```

```
    LOCAL fchsvw[50];                                                   01081
    stvect(stid, $fchsvw);                                               0384
    stvmod($fchsvw, suc);                                                0385
    technm($fchsvw, astr);                                               0386
    RETURN;                                                              0387
    END.                                                                 0388
                                                                         0389
(techno)                                                                 0390
    %This is like TECHNO, but generates the statement number of the
    sub (whether it exists or not).%                                     0391
    %-----------------%                                                  0393
    PROCEDURE(stid, astr);                                               0394
    LOCAL fchsvw[50];                                                   01082
    stvect(stid, $fchsvw);                                               0395
    stvmod($fchsvw, sub);                                                0396
    technm($fchsvw, astr);                                               0397
    RETURN;                                                              0398
    END.                                                                 0399
                                                                         0400
(technm) PROCEDURE (stvwrk, astr);                                       0401
    %This routine will append the statement number to the string,
    given the statement vector.  The address of the first word of the
    statement vector is provided as the first argument, and the
    address of the A-string for the statement number is given as the
    second.                                                              0402
    The algorithm used is roughly as follows:                            0407
        The field of the statement vector corresponding to the           0408
        current level is divided by a base.                              0409
            Two bases are used: 10 (for digits) and 27 (for              0410
            letters and &).                                              0411
        The quotient is added to an appropriate offset to create         0412
        an ascii character, and this character is then added             0413
        to the A-string.                                                 0414
        The division is repeated with the remainder until the            0415
        quotient is less than the base. (This permits statement          0416
        numbers of the form 12AB12D@.)                                   0417
        The base is then changed and the statement vector field          0418
        for the next level is then divided as above.%                    0419
    %-----------------%                                                  0420
    LOCAL                                                                0422
        base, %current base%                                             0423
        basnxt, % next base %                                            0424
        curlev, %current level%                                          0425
        posit, %current plex position%                                   0426
        power, %power of base in plex postion = number of                0427
            characters in statement number for this plex%                0428
        char, %next character in statement number%                       0429
        offset, %converts number indicating position                     0430
            to number/alpha character%                                   0431
        offnxt;  %next offset%                                           0432
    REF astr, stvwrk;                                                    0433
    IF stvwrk = 0 THEN                                                   0565
        BEGIN                                                            0566
        *astr* _ *astr*, '0;                                             0567
        RETURN;                                                          0568
        END;                                                             0569
```

```
   (base, basnxt, offset, offnxt) _                                    0434
      (numbase, alphbase, numoff, alphoff);                            0435
   curlev _ 0;                                                         0437
   LOOP                                                                0438
      BEGIN                                                            0439
      IF (curlev := curlev + 1) >= stvwrk THEN RETURN;                 0440
      posit _ stvwrk[curlev];                                          0441
      %find largest power of base contained in plex position%          0442
      power _ 1;                                                       0443
      WHILE power*base <= posit DO power _ power*base;                 0444
      %now put out the actual string%                                  0445
      UNTIL power < 1 DO                                               0446
         BEGIN                                                         0447
         DIV posit / power, char, posit;                               0448
         *astr* _ *astr*, char + offset;                               0449
         power _ power / base;                                         0450
         END;                                                          0451
      (base, basnxt, offset, offnxt) _                                 0452
         (basnxt, base, offnxt, offset);                               0453
      END;                                                             0454
   END.                                                                0455
(fechux)                                                               0458
   %Given the address of an a-string as the first argument and a file
   number as the second, this routine returns the STID of the
   statement whose statement number is contained in the A-string.
                                                                       0459
   If the statement does not exist (or FECHUX is passed an             0463
   illegal file number or statement number) it returns a -1.           0464
   The algorthm used is roughly as follows:                            0465
      The plex position indicated by each (alphabetic or               0466
      numeric) level field in the statement number is converted        0467
      to an integer, then the psid of the statement                    0468
      corresponding to that position is found.                         0469
      LODCHR is used to read characters from the statement             0470
      number, and the resulting ascii character is converted           0471
      to a number.                                                     0472
      This number is calculated by subtracting an appropriate          0473
      offset (20B for numbers and 40B for lettrs), then                0474
      multiplying the difference by an appropriate base (10            0475
      for digits and 27 for letters).                                  0476
      This number is added to any plex position already                0477
      calculated.                                                      0478
      This subtraction, multiplication, and addition continues         0479
      until a new level in the statement number is                     0480
      encountered.%                                                    0481
   %-----------------%                                                 0482
   PROCEDURE (astr, fileno);                                           0483
   LOCAL stid, %current stid%                                          0484
      curlen, %position current character in statemet number%          0485
      curc, %current character in statemeet number%                    0486
      base, %base for converting character to number%                  0487
      basnxt, %next base%                                              0488
      zero, %zero point for converting character to number%            0489
      zeronxt,  %next zero%                                            0490
      posit; %current plex position%                                   0491
   REF astr;                                                           0492
```

```
   IF fileno NOT IN [0, filmax] THEN RETURN(endfil);                    01354
   posit _ 0;                                                            0493
   stid _ origin;                                                        0494
   stid.stfile _ fileno;                                                 0495
   curlen _ 1;                                                           0496
   base _ numbase;                                                       0497
   basnxt _ alphbase;                                                    0498
   zero _ numoff;                                                        0499
   zeronxt _ alphoff;                                                    0500
   LOOP                                                                  0501
      BEGIN                                                              0502
      LOOP %read characters, calculate position%                         0503
         BEGIN                                                           0504
         curc _ *astr*[curlen];                                          0505
         IF curc IN ['a, 'z] THEN curc _ curc - upcase;                  0506
         % check if have gone from alpha to digit or vice versa          0507
         %                                                               0508
         IF (curc _ curc-zero) NOT IN [0, base) THEN EXIT;               0509
         posit _ posit * base + curc;                                    0510
         IF (curlen _ curlen+1) > astr.L THEN EXIT;                      0511
         END;                                                            0512
      %get stid in position indicated by posit, one level down          0513
      from stid%                                                         0514
      IF posit = 0 THEN RETURN                                           0515
         (IF stid.stpsid = origin THEN stid ELSE endfil);                0570
      IF (stid := getsub(stid)) = stid THEN RETURN (endfil);             0516
      UNTIL (posit _ posit-1) = 0 DO                                     0517
         BEGIN                                                           0518
         IF getftl(stid) THEN RETURN (endfil);                           0519
         stid _ getsuc(stid);                                            0520
         END;                                                            0521
      IF curlen > astr.L THEN RETURN (stid);                             0522
      (base, basnxt, zero, zeronxt, posit) _                             0523
         (basnxt, base, zeronxt, zero, 0);                               0524
      END;                                                               0525
   END.                                                                  0526
                                                                         0527
                                                                         0528
(fechsig) %***%                                                         01332
   PROCEDURE (stid, astrng); %append statement signature to string%
                                                                        01333
   %assumes real statement--not an a-string%                            01334
   %----------------%                                                   01335
   LOCAL sdbadr, word, bytptr, count, char, stdb;                       01336
   REF sdbadr, astrng;                                                  01337
   % eventually want to change this to work for any property type %
                                                                        01338
      IF NOT lodprop( stid, txttyp :&sdbadr, stdb) THEN                 01339
         err($" No text block with this node");                         01340
   % initials %                                                         01341
      word _ getint(stdb);                                              01342
      bytptr _ stbptr(empty) + $word;                                   01343
      count _ 0;                                                        01344
      UNTIL (count _ count + 1) > 4 DO                                  01345
         IF (char _ ^bytptr) = 0 THEN EXIT LOOP                         01346
         ELSE *astrng* _ *astrng*, char;                                01347
```

```
    *astrng* _ *astrng*, SP;                                    01348
  % date and time %                                             01349
    dtfrmt(sdbadr.stime, &astrng);                              01350
  RETURN;                                                       01351
  END.                                                          01352
                                                                01353
FINISH of seggen                                                 0547
```

SIGNALS

```
< NLS, SINTNLS.NLS;9, >, 14-MAR-77 19:28 JDH ;;;;
FILE sintnls % 110 to <rel-nls>sintnls %%(110,) to (rel-nls, sintnls,) %
                                                                      09
   %global refs%                                                    0177
      REF                                                           0178
         tada, subda, cflda, namda, litda, msgda, echoda, ltvsda,
         vspcda;                                                    0179
      REF rawchr, tda;                                              0180
   REGISTER                                                          070
      a1 = 12, %working register%                                    071
      a4 = 15, %working register%                                    072
      r1 = 1,  %working register%                                    073
      r2 = 2,  %working register%                                    074
      r3 = 3,  %working register%                                    075
      r4 = 4,  %working register%                                    076
      r5 = 5,  %working register%                                    077
      rp = 11, %record pointer%                                      078
      p = 7,   %pattern stack%                                       079
      s = 9,   %general call stack pointer%                          080
      m = 10;  %mark stack pointer%                                  081
   (hostnumber)PROC; %get logical host number from tenex%            062
      IF nonetworkflag THEN RETURN(nonetworkflag);                  091
      r1 _ getsbn($"LHOSTN");                                       063
      !JSYS sysgt;                                                  064
      !HRLI r1,0;                                                   065
      !HRR r1,r2;                                                   066
      IF r2 AND SKIP !JSYS getab THEN RETURN(r1);                   067
      RETURN (0);                                                   068
      END.                                                          069
   (tenexverno) PROCEDURE;  %get version number of current TENEX%   032
      %returns number of te form 12900 for tenex version 1.29.00%   033
      RETURN(13301);                                                086
      %the following code was used when NLS ran on a special ARC tenex
      13100 and had to do different things for it:                  087
      LOCAL length, count, table;                                   034
      LOCAL STRING str[70], verno[70];                              035
      LOCAL TEXT POINTER tp1, tp2, tp3, tp4, tp5, tp6;              036
      %%build 6-bit string for sysgt jsys%%                         037
         r1 _ getsbn($"SYSVER");                                    038
      !JSYS sysgt;                                                  039
      length _ -1;                                                  040
      length.RH _ r2.LH;                                            041
      length _ -length-1;                                           042
      table _ r2.RH;                                                043
      count _ 1;                                                    044
      str.L _ length*5;                                             045
      UNTIL (length _ length - 1) < 0 DO                            046
         BEGIN                                                      047
         r1.RH _ table;                                             048
         r1.LH _ count;                                             049
         IF SKIP !JSYS getab THEN                                   050
            str[count] _ r1;                                        051
         count _ count + 1;                                         052
         END;                                                       053
      *verno* _ NULL;                                               054
      IF FIND SF(*str*) [D] ^tp1 _tp1 ['.] ^tp3 ^tp2 _tp2 ['.] ^tp5 ^tp4
```

```
    _tp4 [NOT D] ^tp6 _tp6 THEN                                          055
       *verno* _ tp1 tp2, tp3 tp4, tp5 tp6                               056
    ELSE                                                                 057
       typeas($"tenex version string format changed -- tenexverno
       failed");                                                         058
    IF (length _ VALUE($verno)) > 10000 THEN RETURN(length);             082
    count _ length - ((length/10)*10);                                   083
    RETURN( length _ ((length/10)*100) + count );                        084
    %                                                                    088
    END.                                                                 085
                                                                         061
%the following procedures logically belong to FINTNLS (support routines
near the end) but are here so they will be in the low segment purely for
space considerations.  Any can be moved back to FINTNLS to make a small
increase in the high seg.%                                               092
  (setinterrupts)PROC;                                                  0181
    %set up rubout, command delete, and page write interrupt%            0182
       %IF YOU CHANGE THIS CODE, SEE ALSO (utilty, trapcc) AND
       (utilty, notrapcc)%                                               0183
       chntab[1] _ $stopline .V 1B6; % ^P %                              0184
       %note: the ^C dispatch is altered/restored by the journal%        0185
       chntab[2] _ $ctrlc .V 1B6; %handle ^C to do imlac stuff%          0186
       chntab[3] _ $rubout .V 1B6; % treat ^O like a rubout %            0187
       chntab[4] _ $msgpsi .V 1B6; % for message fork %                  0188
       chntab[9] _ $stkovr .V 1B6;                                       0189
       chntab[11] _ $iodaterr .V 1B6;                                    0190
       chntab[15] _ $ilspsi .V 1B6;  % illegal instruction %            0191
       chntab[17] _ $wrtpsi .V 2B6;                                      0192
       chntab[24] _ $gotohelp .V 1B6;                                    0193
       r1 _ 4B5;                                                         0194
       r2 _ $chntab;                                                     0195
       !HRLI r2, levtab;                                                 0196
       !JSYS sir;                                                        0197
    % special stuff for control-c (^C)%                                  0198
       r1 _ 4B5;                                                         0199
       !JSYS rpcap;                                                      0200
       r3 _ r3 .V 4B11;                                                  0201
       !JSYS epcap; %Permits process to assign ^C for pseudo
       interrupt%                                                        0202
    %enable interrupts%                                                  0203
       r1 _ 4B5;                                                         0204
       !JSYS eir;                                                        0205
    %activate io data error and write pseudo interrupt%                  0206
       r1 _ 4B5;                                                         0207
       r2 _ 105B6; %channel 11, 15, 17%                                  0208
       !JSYS aic;                                                        0209
    %activate rubout, etc.%                                              0210
       r1 _ 20000001B; %^P%                                              0211
       !JSYS ati;                                                        0212
       r1 _ 17000003B; %^O%                                              0213
       !JSYS ati;                                                        0214
       r1 _ 21000030B; %^Q%                                              0215
       !JSYS ati;                                                        0216
       IF nldevice = imlac0 OR nldevice = imlac1 OR nldevice =
       devlproc THEN                                                     0217
          BEGIN                                                          0218
```

```
        r1 _ 3000002B; %^C%                                                  0219
        !JSYS ati;                                                           0220
        r2 _ 340000004000B; %activate rubout, ^P, ^Q, ^C, msgfrk
        term, ^O%                                                            0221
        END                                                                  0222
     ELSE  r2 _ 240000004000B; %activate ^P, ^Q and ^O%                      0223
     r1 _ 4B5;                                                               0224
     !JSYS aic;                                                              0225
  RETURN;                                                                    0226
  END.
                                                                             0227
(gtintmsg) PROCEDURE;                                                        0139
  typeas($"  Message = ");                                                   0140
  LOOP                                                                       0141
     BEGIN                                                                   0142
     !pbin;                                                                  0143
     IF r1 = CA THEN EXIT;                                                   0144
     *intmsg* _ *intmsg*, r1;                                                0145
     END;                                                                    0146
  IF intmsg.L THEN                                                           0147
     BEGIN                                                                   0148
     *intmsg* _ *intmsg*, 0;                                                 0149
     intmsf _ TRUE;                                                          0150
     END;                                                                    0151
  RETURN(intmsg.L);                                                          0152
  END.
                                                                             0153
(setabs)     % new bit table version of settabs %                            0154
  PROCEDURE                                                                  0155
     (tabtbl);       % address of tab table %                               0156
  LOCAL twrd1, twrd2, twrd3;                                                 0157
  REF tabtbl;                                                                0158
     !gjinf();                % get user info %                             0159
     IF r4 = -1 THEN RETURN;          % NOP if detached %                   0160
  % convert NLS internal form to TENEX form %                                0161
     r1 _ tabtbl;                                                            0162
     r2 _ tabtbl[1];                                                         0163
     !SETCA 1,0;              %NLS bits are inverted%                        0164
     !SETCA 2,0;                                                             0165
     twrd1 _ r1;                                                             0166
     twrd2 _ r2;                                                             0167
     r2 _ tabtbl[2];                                                         0168
     !SETCA 2,0;                                                             0169
     twrd3 _ r2;                                                             0170
  % set tabs unless monitor words are the same %                            0171
     !gtabs(18M);                                                            0172
     IF twrd1 = r2 AND twrd2 = r3 AND twrd3 = r4 THEN RETURN                 0173
     ELSE !stabs(18M, twrd1, twrd2, twrd3);                                  0174
  RETURN;                                                                    0175
  END.                                                                       0176
(inittimer) PROCEDURE; %create the timer fork%                                011
  LOCAL jfn;                                                                 012
  LOCAL STRING mfname[20], errstr[200];                                      013
  %chntab[4] contains info for the interrupt routine%                        014
  %activate the timer fork pseudo interrupt%                                 015
     !aic(4B5, 2B10 %channel 4%);                                            016
```

```
   IF NOT continue THEN                                                  017
      BEGIN                                                              018
      IF NOT SKIP !JSYS cfork(0) %create fork% THEN                      019
         err($"cannot create timer fork");                               020
      msgfrk _ r1; %relative fork handle%                                021
      *errstr* _ NULL;                                                   022
      *mfname* _ "MSGNLS"; %must pass name in a writeable string%        023
      IF NOT (jfn _ lgetjfn ($subdir, $mfname, $savext, gtjprf,
      $errstr)) THEN                                                     024
         err($errstr);                                                   025
      r1.LH _ msgfrk;                                                    026
      r1.RH _ jfn;                                                       027
      !get; %get the program mapped in for the fork%                     028
      !epcap(msgfrk, 4B8, 4B8);                                          089
         %allow it to interrupt superior%                                090
      END;                                                               029
   RETURN;                                                               030
   END.
                                                                         031
(idfrmuser) PROCEDURE (str);  %collect ident from user in tty mode%
                                                                         093
   LOCAL char;                                                           094
   REF str;                                                              095
   LOOP                                                                  096
      BEGIN                                                              097
      typeas($"Ident = ");                                               098
      *str* _ NULL;                                                      099
      echon();                                                          0100
      LOOP                                                              0101
         BEGIN                                                          0102
         CASE (char _ rawchr()) OF                                      0103
            = '?: %help him%                                            0104
               BEGIN                                                    0105
               typeas($"
               If you have already been assigned a unique identifier
               (called an IDENT)");                                     0106
               typeas($" by the ARC Identification system (and you
               remember it), ");                                        0107
               typeas($"then please type it followed by a Carriage
               Return.");                                               0108
               typeas($"  Otherwise, just type the ESC (ALTMODE) key
               on your terminal and contact ARC.
               ");                                                      0109
               REPEAT LOOP 2;                                           0110
               END;                                                     0111
            IN ['a, 'z]: char _ char - 40B;                             0112
            IN ['A, 'Z], IN ['2, '6]: NULL;                             0113
            = SP, =BW:  REPEAT LOOP 2;                                  0114
            =CA, =EOL, =CR: EXIT LOOP;                                  0115
            = $ascalt: !haltf;                                          0116
            =BC:                                                        0117
               BEGIN                                                    0118
               IF str.L THEN                                            0119
                  BEGIN                                                 0120
                  typech('/);                                           0121
                  typech(*str*[str.L]);                                 0122
```

```
                    str.L _ str.L - 1;                                  0123
                    END;                                                0124
                 REPEAT CASE;                                           0125
                 END;                                                   0126
              ENDCASE                                                   0127
                 BEGIN                                                  0128
                 typeas($" ? ");                                        0129
                 REPEAT CASE;                                           0130
                 END;                                                   0131
           *str* _ *str*, char;                                         0132
           END;                                                         0133
        IF str.L THEN EXIT LOOP;                                        0134
        END;                                                            0135
     RETURN;                                                            0136
     END.                                                               0137
                                                                        0138
(movsym) PROCEDURE;                                                      02
   % location 116B contains -ln,,sloc where ln is length of symbol table
   and sloc is lowest location used in symbol table.  This procedure
   moves the symbol table so that highest loc used is 377777B and
   updates 116B appropriately.  Symbols can then be defined at runtime
   using the space between new sloc and end of program in low segment. %
                                                                         03
   LOCAL ln;                                                             04
   ln _ -(18M6 .V [116B].LH);                                            05
   mvbtbf([116B].RH, 4B5-ln, ln);                                        06
   [116B].RH _ 4B5-ln;                                                   07
   RETURN END.                                                           08
FINISH                                                                  010
```

SRECORDS

```
< NLS, SRECORDS.NLS;3, >, 21-SEP-76 15:55 KJM ;;;;( MICHAEL,
SRECORDS.NLS;2, ), 16-JUL-74 09:16 EKM ;
FILE srecords % L10  <REL-NLS>srecords %% (L10,) (rel-nls,srecords.rel,)
%                                                                  0263
% records for use with the directory commands %                     02
   (dlinfo) RECORD  % information to be listed for Directory commands %
                                                                    03
      dlidlt[2],        % undeleted or deleted only or both %       04
         % 0 - undeleted only; 1 - deleted only; 2 or 3 - both %   05
      dlifrf[1],        % list for file group (not directory) %     06
      dliacc[1],        % list account of file %                    07
      dliars[1],        % list archive status %                     08
      dliart[1],        % list archive tapes %                      09
      dlidmt[1],        % list dump tape %                          010
      dlidfr[1],        % list default number of versions to keep % 011
      dlilwr[1],        % list last writer %                        012
      dlibyt[1],        % list length and bytesize %                013
      dlimis[1],        % list miscellaneous information about file %
                                                                    014
      dlinrw[1],        % list number of reads and writes %         015
      dliprt[1],        % list protection %                         016
      dlisiz[1],        % list size in pages %                      017
      dlinvr[1],        % don't list vesion numbers %               018
      dlinex[1],        % don't list extension names %              019
      dlitar[2],        % list[time and] date of archiving %        020
         % 1 - date only; 2 or 3 - time and date %                  021
      dlitcr[2],        % list[time and] date of creation this version %
                                                                    022
         % 1 - date only; 2 or 3 - time and date %                  023
      dlitdm[2],        % list[time and] date of last dump %        024
         % 1 - date only; 2 or 3 - time and date %                  025
      dlitov[2],        % list[time and] date original version created %
                                                                    026
         % 1 - date only; 2 or 3 - time and date %                  027
      dlitrd[2],        % list[time and] date of last read %        028
         % 1 - date only; 2 or 3 - time and date %                  029
      dlitwr[2];        % list[time and date of last write %        030
         % 1 - date only; 2 or 3 - time and date %                  031
   (dlgrp) RECORD  % grouping information for Directory commands %  032
      dlgrvr[1],        % group in reverse order %                  033
      dlgacc[1],        % group by accounts %                       034
      dlgars[1],        % group by archive status %                 035
      dlgdar[1],        % group by archive date %                   036
      dlgart[1],        % group by archive tapes %                  037
      dlgbyt[1],        % group by bytesize %                       038
      dlgdcr[1],        % group by creation date %                  039
      dlgdlt[1],        % group by deleteion status %               040
      dlgddm[1],        % group by dump date %                      041
      dlgdmt[1],        % group by dump tapes %                     042
      dlgdfr[1],        % group by file retention specs %           043
      dlglwr[1],        % group by last writer %                    044
      dlgdov[1],        % group by creation date of original version %
                                                                    045
      dlgprt[1],        % group by protection %                     046
      dlgdrd[1],        % group by read date %                      047
      dlgdwr[1];        % group by write date %                     048
```

```
(dlsort) RECORD  % sorting information for Directory commands %   049
   dlsrvr[1],          % sort in reverse order %                  050
   dlsacc[1],          % sort by accounts %                       051
   dlsalp[1],          % sort alphabetically %                    052
   dlsart[1],          % sort by archive tapes %                  053
   dlstar[1],          % sort by archive time and date %          054
   dlsbyt[1],          % sort by bytesize %                       055
   dlstcr[1],          % sort by time and date of creation %      056
   dlsdlt[1],          % sort by deletion status %                057
   dlstdm[1],          % sort by time and date of dump %          058
   dlsdmt[1],          % sort by dump tape %                      059
   dlsdfr[1],          % sort by file retention specs %           060
   dlslwr[1],          % sort by last writer %                    061
   dislen[1],          % sort by length in bytes %                062
   dlsnac[1],          % sort by number of accesses %             063
   dlsnrd[1],          % sort by number of reads %                064
   dlsnwr[1],          % sort by number of writes %               065
   dlstov[1],          % sort by orig. version creation time & date %
                                                                  066
   dlstrd[1],          % sort by last read time and date %        067
   dlssiz[1],          % sort by size in pages %                  068
   dlstwr[1];          % sort by last write time and date %       069
%Record Definitions%                                             070
(blkhdr)     % **24 BIT words** block header record%              071
   RECORD                                                         072
   %NOTE IF YOU CHANGE THIS, ALSO CHANGE RECORD DLMBLK BELOW%     073
   blklength[11],   %length of block requested%                   074
   sysblength[11],  %actual length of block%                      075
   blkfree[1],     %true if block is on a free list%              076
   blkprevfree[1]; %true if previous block is on a free list%     077
                                                                  078
                   %only the first word is valid for a block
                   that is in use%                                079
                                                                  080
(cabts)      %content analysis bits%                             084
   RECORD                                                         085
   cackf[1],    %true if have tested with current pattern%        086
   capsf[1];    %true if have passed with current pattern.%       087
(chars) RECORD                                                   088
   chnul[1],                                                      089
   chr4[7],                                                      0348
   chr3[7],                                                      0347
   chr2[7],                                                      0346
   chr1[7],                                                      0345
   chr0[7];
                                                                 0344
(displayarea) RECORD %entrylength = dal, max = damax%            0103
   % **** IF YOU CHANGE THIS RECORD DEFINITION, FOR SURE TELL THE
   OUTPUT PROCESSOR GUY SO HE CAN CHANGE THE COPY IN THE OUTPUT
   PROCESSOR -- BEFORE YOU BRING UP A NEW NLS!!! **** %          0104
   davspec[36],  % SEQUENCE  first viewspec word--1st word in record%
                                                                 0105
                                                                 0106
   davspc2[36],  % SEQUENCE second viewspec word--2nd word in record%
                                                                 0107
                                                                 0108
```

```
    dapvs[36],     % SEQUENCE previous first viewspec word%          0109
                                                                     0110
    dapvs2[36],    % SEQUENCE previous second viewspec word%         0111
                                                                     0112
    dacsp[36],     % SEQUENCE csp for this da%                       0113
                                                                     0114
    daccnt[12],    % SEQUENCE character count for dacsp t-pointer%   0115
    dacsize[2],    % FORMATTER character size%                       0116
    danocs[2],     % FORMATTER horizontal increment%                 0117
    dasgcs[2],     % FORMATTER horizontal increment%                 0118
    dafrzl[18],    % SEQUENCE a frozen statement chain for this da%
                                                                     0119
                                                                     0120
    daind[18],     % FORMATTER indentation per level%                0121
    damind[18],    % FORMATTER max indentation%                      0122
                                                                     0123
    dalftjst[18], % ???? left-justify branch, plex for indentation
    off%                                                             0124
    dacrow[18],    % FORMATTER current row count%                    0125
                                                                     0126
    damrow[18],    % FORMATTER max row count%                        0127
    daccol[18],    % FORMATTER current column count%                 0128
                                                                     0129
    damcol[18],    % FORMATTER max column count%                     0130
    dahinc[18],    % FORMATTER horizontal increment%                 0131
                                                                     0132
    davinc[18],    % FORMATTER vertical increment%                   0133
    danohi[18],    % FORMATTER horizontal increment%                 0134
                                                                     0135
    dasghi[18],    % FORMATTER horizontal increment%                 0136
    dafont[18],    % FORMATTER default font%                         0137
                                                                     0138
    dalink[18],    % ???? a link-jump stack for this da%             0139
    dahandle[9],   % FRONTEND system handle for this display area%   0140
    dalssup[9],    % FRONTEND next ls number to suppress for literal
    feedback%                                                        0141
                                                                     0142
    daleft[18],    % FRONTEND left boundry of da (raster units)%     0143
    daright[18],   % FRONTEND right boundry of da (raster units)%    0144
                                                                     0145
    datop[18],     % FRONTEND top boundry of da (raster units)%      0146
    dabottom[18], % FRONTEND bottom boundry of da (raster units)%    0147
       %(0,0) is upper left corner%                                  0148
                                                                     0149
    dalsrt[18],    % FORMATTER LSRT starting address%                0150
    dalsz[9],      % FORMATTER LSRT size for this da%                0151
    dapstf[5],     %file id, previous value for this da%            0152
    dashrinkcnt[2], % FORMATTER LSRT allocation shrink algorithm cnt%
                                                                     0153
    daexis[1],     %this da entry is in use%                        0154
    daempty[1],    %da-has-no-display flag%                         0155
                                                                     0156
    datab0[36],    % FORMATTER first tab position word%             0157
    datab1[36],    % FORMATTER second tab position word%            0158
    datab2[36],    % FORMATTER third tab position word%             0159
                                                                     0160
```

```
   dacacode[18],   % SEQUENCE address of content analyzer program --
   or 0 %                                                          0161
   dausqcod[18],   % SEQUENCE address of sequence generator program
   -- or 0 %                                                       0162
                                                                   0163
   dalhandle[18],   % FRONTEND 0 or handle for linked da%          0164
   dalsidb[18],      % FRONTEND address of lsid bit table %        0165
                                                                   0166
   daukeycod[18],   % FRONTEND  address of sort key extractor program
   -- or 0 %                                                       0167
   daauxiliary[1], % FRONTEND a da used for other purposes than
   displaying.
      (as done in TNLS)  Calculator, for example, uses such an area.%
                                                                   0168
   dafrozen[1], % FRONTEND frozen boundaries flag (calculator)%    0169
   daseq[1],     % FRONTEND sequential display area%               0170
   dalneighbor[3],     % FRONTEND True if window has a left neighbor%
                                                                   0171
   darneighbor[3],     % FRONTEND True if window has a right
   neighbor%                                                       0172
   datneighbor[3],     % FRONTEND True if window has a top neighbor%
                                                                   0173
   dabneighbor[3],     % FRONTEND True if window has a bottom
   neighbor%                                                       0174
   dasuppress[1];   % FRONTEND the display image is suppressed%   0175
   DECLARE EXTERNAL                                                0176
      %if you enlarge damax, yo must enlarge
      dalneighbor,...,dabneighbor also%                            0177
         dal= 22,                                                  0349
         damax= 7;                                                 0350
(frrentry) RECORD  %file return ring entry%                        0178
   frsrring[18],  %address of statement return ring%               0179
   frexis[1];     %true if exists%                                 0180
   DECLARE EXTERNAL                                                0181
      frrelen= 1;  %length of file return ring header%             0351
(frrheader) RECORD   %file return ring header%                     0182
   frhtop[6],  %index to top of ring%                              0184
   frhlast[6], %index to last entry%                               0185
   frhexis[1]; %true if valid frr header%                          0183
   DECLARE EXTERNAL                                                0186
      frrhlen= 1;  %length of file return ring header%             0352
(gbhst) RECORD        % word returned by getab from HOSTN table %  0187
   hstnmi[18],        % index into HSTNAM table %                  0188
   hstnmn[8],         % host number for this entry %               0189
   hstnmf[10]         % flags for this host %                      0190
   ;                                                               0191
(lsrt) RECORD  %line segment reference table entry%                0192
   rtbps[36],     %starting byte pointer to text - increment and
   load byte gets 1st byte of string%                              0193
   rtbpe[36],     %ending byte pointer to text - plus 1 char further%
                                                                   0194
   rtstid[36],    %stid of statement being displayed%             0195
   rtx1[18],      %x-cord position at start of line segment%      0196
   rtx2[18],      %last character, or last plus 1?%               0197
                  %rtx2 also used as stop coord for formatter%    0198
   rty[18],       %y-cord%                                        0199
```

```
   rtlev[6],      %level of this statement%                          0200
   rtccnt[12],    %ordinal relative to start of stmt of first char of
   line segment%                                                     0201
   rthinc[14],    %horizontal increment%                             0202
   rtlsid[9],     %system handle (jsys return) %                     0203
   rtexis[1],     %entry exists flag %                               0204
   rtsrce[4],     %source of display data (code)%                    0205
   rtfont[5],     %font of this entry%                               0206
   rtcbug[3],     %#chars which constitute 1 buggable char%          0207
                                                                     0208
   rtlplink[12],  %Link-list pointer for line processor display use%
                                                                     0209
   rtpartst[1],   %partial statement, on for last line seg of stid%
                                                                     0210
   rtnew[1],      %new entry flag used to recognize changes%         0211
   rtcsize[2],    %character size%                                   0212
   rtllsg[1];     %last line segment on the line flag%
                                                                     0213
   DECLARE EXTERNAL                                                  0214
      lsrtl= 7;
                                                                     0353
(srrentry) RECORD  %statement return ring entry%                     0215
   srpsid[18],  %partial statement pointer for a statement%          0216
      %to be combined with srhfileno to form stid%                   0217
   srcc[12],    %character count%                                    0218
   srexis[1],   %true if exists%                                     0219
   srvs1[36],   %first viewspec word%                                0220
   srvs2[36];   %second viewspec word%                               0221
   DECLARE EXTERNAL                                                  0222
      srrelen= 3;  %length of file return ring entry%                0354
(srrheader) RECORD  %header for statement return ring%               0223
   srhname[18],  %address of file name string%                       0224
   srhfileno[5],  %nls file number for this file %                   0225
      % 0 if file not open%                                          0226
   srhexis[1],     %true if valid frr header%                        0227
   srhtop[6],      %index to current top of ring%                    0228
   srhlast[6];     %index to last entry%                             0229
   DECLARE EXTERNAL                                                  0230
      srrhlen= 1;  %length of statement return ring header%          0231
(viewspecs) RECORD                                                   0319
   % **** IF YOU CHANGE THIS RECORD DEFINITION, FOR SURE TELL THE
   OUTPUT PROCESSOR GUY SO HE CAN CHANGE THE COPY IN THE OUTPUT
   PROCESSOR -- BEFORE YOU BRING UP A NEW NLS!!! **** %              0320
   vslev[6],      %lower level bound--level clipping%                0321
   vstrnc[6],     %line truncation value%                            0322
   vsrlev[6],     %relative level%                                   0323
   vslevd[1],     %direction of relative level adjustment%           0324
   vscapf[1],     %content analyzer pass-flag%                       0325
   vsusqf[1],     %user sequence generator flag%                     0326
   vsbrof[1],     %branch only flag%                                 0327
   vsplxf[1],     %plex-only flag%                                   0328
   vsblkf[1],     %blank line flag%                                  0329
   vsindf[1],     %indenting on flag%                                0330
   vsnamf[1],     %names on flag%                                    0331
   vsstnf[1],     %loc-nums on flag%                                 0332
   vsstnr[1],     %loc-nums on right flag%                           0333
```

```
    vsaflf[1],     %abbreviated feedback line flag%                    0334
    vsrind[1],     %relative indenting flag%                           0335
    vsfrzf[1],     %frozen statements flag%                            0336
    vscakf[1],     %content analyzer k viewspec flag%                  0337
    vsidtf[1],     %initials, date, and time flag%                     0338
    vspagf[1],     %page flag for tty output%                          0339
    vsdaft[1],     %display area format flag%                          0340
    vssidf[1];     %display sids flag%                                 0341
(view2specs) RECORD                                                    0342
    vsmkrf[1];     %display markers flag%                              0343
(iptmchr) RECORD %time for remote display%                             0281
    tmchr6[6], %low order 6 bits%                                      0282
    tmchr5[6],                                                         0283
    tmchr4[6],                                                         0284
    tmchr3[6],                                                         0285
    tmchr2[6],                                                         0286
    tmchr1[6]; %high order 6 bits%                                     0287
(errordoc) RECORD  %for recording line processor errors -- used by
,inpfbk,errfill and user subsystem lpinterp%                           0291
    ercode[36], % code for type of record %                            0292
    erhost[36], % host number %                                        0293
    eruser[36], % user name %                                          0294
    ercondir[36], % connected directory %                              0295
    erline[18], % tty line number %                                    0296
    erjobn[18], % TENEX job number %                                   0297
    ertipn[36], % TIP number if any %                                  0298
    erdatime[36], % time and date %                                    0299
    erlpbaudf[18], % baud rate factor %                                0300
    erlptype[18], % LP type code %                                     0301
    erprom[36], %LP prom number%                                       0302
    erfns1[36], % file name string %                                   0303
    erfns2[36],                                                        0304
    erfns3[36],                                                        0305
    erfns4[36],                                                        0306
    erfns5[36],                                                        0307
    erfns6[36],                                                        0308
    erdlen[36], % number of data items %                               0309
    erdata[36]; % first data word %                                    0310
(octchar) RECORD %To get at the octal representations of a character%
                                                                       0311
    ocjnk1[18],                                                        0312
    oc2qtr[9],  %2nd quarter of a word%                                0313
    ocjnk2[2],                                                         0314
    ochar3[3],  %3rd char of octal representation of a char%           0315
    ochar2[3],  %2nd char ....%                                        0316
    ochar1[1];  %1st char ....%                                        0317
FINISH                                                                 0262
% We think these are not needed%                                      0264
(ckpbits) RECORD % checkpoint bits %                                   0265
    ckpt1[1],    % need to update first checkpoint file %            0266
    ckpt2[1],    % need to update second checkpoint file %           0267
    nxckpt[1];   % which checkpoint to update next %                  0268
(halfword) RECORD                                                      0269
    rhword[18], %right half word%                                      0270
    lhword[18]; %left half word%
                                                                       0271
```

```
(inpsent) RECORD %format of input stack element%                    0272
   ipstid[36],  %stid, if marker, else endfil%                      0273
   ipchar[36],  %last character read, (char count, if marker)%      0274
   ipcoords[36],  %coordinates of last character read%              0275
   iptime[36];  %time charactr read%
                                                                    0276
   DECLARE EXTERNAL inpsel = 4;  %length of element on input stack%
                                                                    0277
(inrptrs) RECORD %pointers to input element%                        0278
   inpcur[18],  %last read by lookc%                                0279
   inpptr[18];  %last read by input%
                                                                    0280
(reghalves) RECORD %for using two halves of words%                  0288
   rh[18], %right half word%                                        0289
   lh[18]; %left half word%
                                                                    0290
```

STGMGT

```
< NLS, STGMGT.NLS;38, >, 19-DEC-74 19:08 JDH ;;;;<NLS>STGMGT.NLS;38,
28-MAR-74 14:44 HGL ;
FILE stgmgt    % L10  <rel-NLS>stgmgt %% (L10,) (rel-nls,stgmgt.rel,) %
                                                                      0294
DECLARE %zone format%                                                 0268
   wrdsused = 0, blkmax = 1,  blkmin = 2,  blkmxi = 3,   blksused = 4,
   zonend = 5, zfrelst = 6;                                           0269
                                                                      0270
%storage allocator%                                                    02
   (getblk) PROCEDURE  %allocate a blk of "size" words in "zone"%      03
      (size,  %number of wods desired in blk%                         04
      zone);  %address of free storage zone%                          05
      %------------------%                                            06
      %allocates a blk in zone of at least size+bhl words.  The blk
      consists of a blk header and at least sizee words of zeroes.  The
      address returned is that of the begining of the blk (the blk
      header).  There are at least size usable words at BlockAddress +
      bhl.  If no blk can be allocated, a zero is returned.%           07
      %------------------%                                            08
      LOCAL                                                           09
         blk,     %address of blk%                                    010
         r,       %remainder temp%                                    011
         zero,    %temp%                                              012
         end,     %temp%                                              013
         index;   %index into zones free lists for blocks of
                   correct size%                                      014
      REF blk, zone;                                                  015
      BUMP cgetblk;                                                   016
      IF zone[blkmin] = zone[blkmax] THEN                             017
         IF zone[blkmax] >= size+bhl THEN                             018
            index _ 0                                                 019
         ELSE                                                         020
            RETURN(0)                                                 021
      ELSE                                                            022
         BEGIN  %round up to nearest available size%                  023
         index _ (size + bhl + zone[blkmin] - 1)/zone[blkmin] - 1;    024
         IF index NOT IN [0, zone[blkmxi]] THEN RETURN(0);            025
         END;                                                         026
      IF NOT zone[zfrelst + index] AND (zone[blkmin] = zone[blkmax] OR
      NOT replenish(index, &zone)) THEN                               027
         %no free blocks of that index or larger%                     028
         RETURN(0);                                                   029
      &blk _ zone[zfrelst + index];                                   030
      %unlink it%                                                     031
         unlink(&blk, &zone);                                         032
      %zero the blk%                                                  033
         zero _ &blk + bhl;                                           034
         end _ &blk + blk.sysblength;                                 035
         DO [zero] _ 0 UNTIL (zero _ zero + 1) >= end;                036
      blk.blklength _ size+bhl; %for caller's use only%               037
      BUMP zone[blksused];                                            038
      zone[wrdsused] _ zone[wrdsused] + blk.sysblength;               039
      RETURN(&blk);                                                   040
      END.                                                            041
                                                                      042
   (treeblk) PROCEDURE  %de-allocate a block in zone%                 043
```

```
   (blk,  %address of block to be de-allocated%                          044
   zone); %address of free storage zone%                                 045
   %----------------%                                                    046
   %the block addressed by blk is marked as free and linked back into
   the appropriate free list in zone.                                    047
   NOTE:  Freeblk will return FALSE if it finds a bad block address.
   You may run out of space if you never issue another good free to
   replace a bad call.%                                                 0385
   %----------------%                                                    048
   LOCAL                                                                 049
      previous, %address of previous block in the zone%                  050
                %for combining free blocks%                              051
      next;     %address of next block in zone%                        052
                %for combining free blocks%                              053
   REF blk, previous, next, zone;                                        054
   BUMP cfreeblk;                                                        055
   IF &blk NOT IN [&zone, zone[zonend]] THEN                             056
      RETURN(FALSE);                                                     058
   zone[wrdsused] _ zone[wrdsused] - blk.sysblength;                     061
   IF zone[blkmin] NOT= zone[blkmax] THEN                                062
      BEGIN                                                              063
      &next _ &blk + blk.sysblength;                                     064
      IF &next < zone[zonend] THEN                                       065
         BEGIN                                                           066
         IF next.blkfree AND (blk.sysblength + next.sysblength <=
         zone[blkmax]) THEN                                              067
            BEGIN %combine them%                                         068
            unlink(&next, &zone);                                        069
            blk.sysblength _ blk.sysblength + next.sysblength;           070
            &next _ &blk + blk.sysblength;                               071
            END;                                                         072
         END;                                                            073
      IF blk.blkprevfree AND (blk.sysblength + [&blk - 1] <=
      zone[blkmax]) THEN                                                 074
         BEGIN %combine them%                                            075
         %last word of previous block contains the length of that
         block%                                                          076
         &previous _ &blk - [&blk - 1];                                  077
         unlink(&previous, &zone);                                       078
         previous.sysblength _ previous.sysblength + blk.sysblength;
                                                                         079
         &blk _ &previous;                                               080
         END;                                                            081
      END;                                                               082
   linkup(&blk, &zone);                                                  083
   BUMP DOWN zone[blksused];                                             084
   RETURN(TRUE);                                                         085
   END.
                                                                         086
(makezone) PROCEDURE  %make a zone out of a block%                       087
   (zone,  %address of a block to be initialized as a zone%              088
   max,    %maximum size of a block -- integral multiple of min%         089
   min,    %minimum size block%                                          090
   size);  %size of the zone%                                            091
   %----------------%                                                    092
   %This routine takes the address of a free storage block and
```

```
   constructs an appropriate header so that it will serve as a free
   storage zone.  If min and max are the same, then the zone will
   contain only fixed sized block and a single free list -- no
   combining of adjacent free blocks will be attempted by freeblk%
                                                                    093
   %----------------%                                               094
   LOCAL                                                            095
      blk,     %address of a block within the zone%                 096
      lastfreelist,    %displacement to the lastfreelist free list
      for this zone%                                                097
      frstblk, %address of first block in the zone%                 098
      base,    %address of a block within the zone%                 099
      length;  %length of rest of block%                           0100
   REF zone, blk, frstblk, base;
                                                                   0101
   % zero out this whole block before starting %                   0386
      FOR length _ (size-1) DOWN UNTIL < 0 DO [&zone + length] _ 0;
                                                                   0387
   BUMP cmakezone;                                                 0102
   max _ ((max+min-1)/min)*min;                                    0103
   zone[blkmax] _ max;                                             0104
   zone[blkmin] _ min;                                             0105
   zone[blkmxi] _ (max/min) - 1;                                   0106
   lastfreelist _ zone[blkmxi] + zfrelst;                          0107
   &frstblk _ &base _ &zone + lastfreelist + 1;                    0108
   length _ ((size - (&frstblk - &zone))/min)*min;                 0109
   zone[zonend] _ &frstblk + length;                               0110
   UNTIL length < max DO                                           0111
      BEGIN                                                        0112
      &blk _ &base + max;                                          0113
      base.sysblength _ max;                                       0114
      [&blk-1] _ max;                                              0115
      base.blkfree _ base.blkprevfree _ TRUE;                      0116
      base[blksuc] _ zone[lastfreelist] := &base;                  0117
      base[blkprd] _ 0;                                            0118
      IF &base _ base[blksuc] THEN                                 0119
         base[blkprd] _ zone[lastfreelist];                        0120
      &base _ &blk;                                                0121
      length _ length - max;                                       0122
      END;                                                         0123
   frstblk.blkprevfree _ FALSE;                                    0124
   IF length > 0 THEN                                              0125
      BEGIN                                                        0126
      blk.sysblength _ length;                                     0127
      IF length < min THEN                                         0128
         blk.blkfree _ blk.blkprevfree _ FALSE                     0129
      ELSE                                                         0130
         BEGIN                                                     0131
         blk.blkfree _ blk.blkprevfree _ TRUE;                     0132
         blk[blksuc] _ zone[zfrelst + length/min - 1] := &blk;     0133
         blk[blkprd] _ 0;                                          0134
         IF &base _ blk[blksuc] THEN                               0135
            base[blkprd] _ &blk;                                   0136
         [&blk + blk.sysblength -1] _ length;                      0137
         END;                                                      0138
      END;                                                         0139
```

```
   zone[blksused] _ zone[wrdsused] _ 0;                                  0140
   RETURN;                                                               0141
   END.                                                                  0142
                                                                         0143
(replenish) PROCEDURE %replenish supply of free blocks in zone%          0144
   (index, %free list index for size blocks needed%                      0145
   zone);  %address of free storage zone%                                0146
   %--------------%                                                      0147
   %This routine will try to find a larger block which can be
   subdevided to replenish the supply of blcks desired.  First it
   will try to find a block which is a multiple in size (so all
   pieces will go on same free list).  If this fails, a scan for any
   larger block is made.  If non is found, it returns FALSE (could be
   made to combine smaller blocks as well), otherwise, breakup is
   used to decompose the larger block and add to the supply of
   desired blocks and this routine returns TRUE%                         0148
   %--------------%                                                      0149
   LOCAL                                                                 0150
      indexplus1, %value of index+1 (used to speed up loop)%             0151
      blk,   %address of a larger free block%                            0152
      maxi,  %value of zone[blkmxi] (used to speed up loop)%             0153
      n,     %temp, used in loop%                                        0154
      i;     %temp, used as loop variable ad free list index%            0155
   REF blk, zone;                                                        0156
   BUMP creplenish;                                                      0157
   IF index NOT IN [0, zone[blkmxi]] THEN                                0158
      BEGIN                                                              0290
      dismes(2,$" index bad, replenish");                                0291
      RETURN(FALSE);                                                     0292
      END;                                                               0293
   maxi _ zone[blkmxi];                                                  0163
   %try for multiples first%                                             0164
      i _ indexplus1 _ index+1;                                          0165
      n _ 2;                                                             0166
      UNTIL (i _ indexplus1*n - 1) > maxi DO                             0167
         BEGIN                                                           0168
         IF zone[i + zfrelst] THEN %got one%                             0169
            BEGIN                                                        0170
            unlink(&blk _ zone[i+zfrelst], &zone);                       0171
            breakup(&blk, &zone, index);                                 0172
            RETURN(TRUE);                                                0173
            END;                                                         0174
         BUMP n;                                                         0175
         END;                                                            0176
   %Find any%                                                            0177
      i _ index;                                                         0178
      UNTIL (i _ i + 1) > maxi DO                                        0179
         IF zone[zfrelst + i] THEN %got one%                             0180
            BEGIN                                                        0181
            unlink(&blk _ zone[zfrelst+i], &zone);                       0182
            breakup(&blk, &zone, index);                                 0183
            RETURN(TRUE);                                                0184
            END;                                                         0185
   %should now try to combine smaller adjacent blocks to form one
   that is large enough -- so far we have not needed this.  Write it
   when we need it%                                                      0388
```

```
   RETURN(FALSE);                                                    0186
   END.                                                              0187
                                                                     0188
(breakup) PROCEDURE  %break up blk and put in index free list%      0189
   (blk,    %address of block to be subdivided%                     0190
   zone,    %address of free storage zone%                          0191
   index);  %free list index to be replenished%                     0192
   %-----------------%                                              0193
   %This routine creates a number of blocks (by breaking up blk) of
   appropriate size for the free list indicated by index and links
   them into that free list in zone.%                               0194
   %-----------------%                                              0195
   LOCAL                                                            0196
      length, %length of smaller blocks%                            0197
      base,   %temp, used in subdividing blk%                       0198
      i,      %loop control variable%                               0199
      r;      %temp, used to set up i%                              0200
   REF blk, base, zone;                                             0201
   length _ (index+1)*zone[blkmin];                                 0202
   DIV blk.sysblength/length, i, r;                                 0203
   IF NOT r THEN BUMP DOWN i;                                       0204
   UNTIL (i _ i -1) < 0 DO                                          0205
      BEGIN                                                         0206
      &blk _ (&base _ &blk) + length;                               0207
      blk.sysblength _ (base.sysblength := length) - length;        0208
      linkup(&base, &zone);                                         0209
      END;                                                          0210
   &blk _ (&base _ &blk) + blk.sysblength;                          0211
   IF base.sysblength >= zone[blkmin] THEN                          0212
      linkup(&base, &zone)                                          0213
   ELSE base.blkfree _ FALSE;                                       0214
   RETURN;                                                          0215
   END.                                                             0216
                                                                    0217
(unlink) PROCEDURE  %unlink a block from its free list%             0218
   (blk,  %address of block to be unlinked from its free list%      0219
   zone); %address of a free storage zone%                          0220
   %--------------%                                                 0221
   %this routie unlinks blk from the appropriate free list in zone%
                                                                    0222
   %--------------%                                                 0223
   LOCAL                                                            0224
      next, %address of next block in zone%                         0225
      suc,  %address of block pointed to by successo link%          0226
      prd;  %address of block pointed to by predecessor link%       0227
   REF  next, blk, suc, prd, zone;                                  0228
   BUMP cunlink;                                                    0229
   blk.blkfree _ FALSE;                                             0230
   IF (&next _ &blk + blk.sysblength) < zone[zonend] THEN           0231
      next.blkprevfree _ FALSE;                                     0232
   IF &suc _ blk[blksuc] THEN                                       0233
      suc[blkprd] _ blk[blkprd];                                    0234
   IF &prd _ blk[blkprd] THEN                                       0235
      prd[blksuc] _ blk[blksuc] := 0                                0236
   ELSE                                                             0237
      zone[zfrelst + blk.sysblength/zone[blkmin] - 1] _ blk[blksuc]
```

```
      := 0;                                                          0238
   blk[blkprd] _ 0;                                                  0239
   RETURN;                                                           0240
   END.                                                              0241
                                                                     0242
(linkup) PROCEDURE  %linkup blk into appropriate free list%          0243
   (blk,  %address of block to be linked into free list%             0244
   zone); %address of a free storage zone%                           0245
   %--------------%                                                  0246
   %this routie links blk into the begining of the appropriate free
   list in zone%                                                     0247
   %--------------%                                                  0248
   LOCAL                                                             0249
      next,   %address of next block in zone%                        0250
      index;  %free list index for blk%                              0251
   REF  next, blk, zone;                                             0252
   BUMP clink;                                                       0253
   blk.blkfree _ TRUE;                                               0254
   index _ blk.sysblength/zone[blkmin]-1 + zfrelst;                  0255
   blk[blksuc] _ zone[index] := &blk;                                0256
   blk[blkprd] _ 0;                                                  0257
   [&blk+blk.sysblength-1] _ blk.sysblength;                         0258
   IF (&next _ &blk+blk.sysblength) < zone[zonend] THEN              0259
      next.blkprevfree _ TRUE;                                       0260
   IF &blk _ blk[blksuc] THEN                                        0261
      blk[blkprd] _ zone[index];                                     0262
   RETURN;                                                           0263
   END.                                                              0264
                                                                     0265
%support routines to control storage allocator%                      0298
   %string and array handling%                                       0328
   (getstring) PROCEDURE (length, zone);                             0329
      %allocate a block in "zone" of  "length" characters  plus
      block header plus string header.  Initialize the string to
      empty.  Return string address.                                 0330
      length       actual useable length of string desired           0331
      zone         address of zone with free space in it             0332
      SIGNALS if no more room in that block%                         0361
      LOCAL blockadress;                                             0334
      REF blockadress;                                               0340
      &blockadress _ getblk ((length + 4)/5 + bhl                    0335
         + %string header% 1, zone);                                 0360
      IF NOT &blockadress THEN SIGNAL (errsig, $"Fatal storage
      shortage error");                                              0342
      &blockadress _ &blockadress + bhl;                             0354
      blockadress.M _ length;                                        0339
      *blockadress* _ NULL;                                          0341
      RETURN (&blockadress);                                         0343
      END.                                                           0344
                                                                     0345
   (getarray) PROCEDURE (length, zone);                              0390
      %allocate an array in "zone" of  "length" words  plus  block
      header.  Initialize the array to empty.  Return array address.
                                                                     0391
      length       actual useable length of array desired            0392
      zone         address of zone with free space in it             0393
```

```
      SIGNALS if no more room in that block%                                0394
      LOCAL blockadress;                                                    0395
      REF blockadress;                                                      0396
      &blockadress _ getblk (length + bhl, zone);                           0397
      IF NOT &blockadress THEN SIGNAL (errsig, $"Fatal storage
      shortage error");                                                     0398
      RETURN (&blockadress _ &blockadress + bhl);                           0399
      END.                                                                  0400
                                                                            0401
   (freestring) PROCEDURE (straddr, zone); % also for arrays. %             0347
      %Deallocate a string at "straddr" within "zone".  Returns FALSE
      if no such block in "zone"%                                           0348
      RETURN(freeblk (straddr - bhl %string header%, zone));                0349
      END.                                                                  0350
                                                                            0351
%special display storage problems%                                          0299
   (maklsrt) PROCEDURE (zone, segs, da);                                    0362
      %allocate a new block in "zone" to be used as a line segment
      reference table                                                       0363
      updates da record                                                     0364
      returns FALSE if there was no more room in "zone", ELSE address
      ot first useable location (past header)                               0365
                                                                            0366
      zone        address of start of zone containing free space            0367
      segs        number of segs possible in this display area              0368
      da          address of display area%                                  0369
                                                                            0370
      LOCAL lsrtaddress, words;                                             0371
      REF da;                                                               0372
      IF da.dalsrt OR nlmode # fulldisplay THEN RETURN (da.dalsrt);
      %no-op if already exists%                                             0373
      words _ MIN(segs * lsrtl, 777B);                                      0374
      lsrtaddress _ getblk (words, zone );                                  0375
      IF lsrtaddress THEN                                                   0376
         BEGIN                                                              0377
         lsrtaddress _ lsrtaddress + bhl; %block header%                    0378
         da.dalsz _ words/lsrtl;                                            0379
         da.dalsrt _ lsrtaddress;                                           0380
         <DSPGEN, clrall>(&da, FALSE);%clear - no strings to free%
                                                                            0381
         END;                                                               0382
      RETURN (lsrtaddress);                                                 0383
      END.
                                                                            0384
   (freelsrt) PROCEDURE (zone, da);                                         0319
      %deallocates a line segment reference table and updates da
      record                                                                0321
      zone              zone address                                        0322
      da               address of display area record%                    0323
      REF da;                                                               0324
      IF NOT da.dalsrt OR nlmode # fulldisplay THEN RETURN; %no-op if
      none exists%                                                          0355
      freeblk(da.dalsrt - bhl, zone);                                       0325
      da.dalsz _ da.dalsrt _ 0;                                             0326
      RETURN END.                                                           0327
FINISH                                                                      0288
```

SYNTAX

```
SYNTAX >>> < NLS, SYNTAX.NLS.53, >, 16-Jan-78 13:53 HGL ;;;; index in
<nls,sysgd,>
FILE nislanguage% CML <rel-nls>SYNTAX % % ( CML,) (rel-nls,SYNTAX.rel,)
%                                                                    02
% DECLARATIONS %                                                    024
   DECLARE PARSEFUNCTION                                            025
      answ,          % reads answer construct %                     026
      answer,        % for questions - returns  0/1 %             02753
      lookstat,      % looks for s and shows status for substitute command
      %                                                           04254
      sp,            % reads next char, TRUE if space %             027
      readconfirm,   % reads next char if ca %                     028
      myrdconfirm,   % reads next char if ca; with variations for HELP %
                                                                  04219
      dumprompt,                                                  05033
      readbug,       % reads next char if BUG %                   02003
      readoption,    % TRUE if next char is optchar %              029
      readrepeat,    % TRUE if next char is repeat %               030
      lookansw,      % TRUE if next char is Y/CA %                 031
      sublookansw,   % TRUE if next char is Y/CA with variations for
      substitute command %                                        04260
      lookconfirm,   % TRUE if next char is CA/REPEAT/INSERT %    02754
      lookbug,       % TRUE if next char is BUG %                 02004
      mylookbug,     % TRUE if next char is BUG with variations for HELP %
                                                                  03982
      lookback,      % TRUE if next char is _; for HELP %         03980
      looknum,       % TRUE if next char is a number %              032
      lookrpt,       % TRUE if next char is rpt; for HELP %       03981
      prmptlsel,     % prompt for an LSEL %                       02311
      prmptdsel,     % prompt for an DSEL %                       03842
      prmptssel,     % prompt for an SSEL %                       03841
      notca;         % reads next char, TRUE iff not CA char %      033
   DECLARE EXTERNAL zinsstatement, textent, text1, structure, cshmode,
   nlssubs, synsubs, allsubs, prot1, devopt, prtgrp, qdhelp, qthelp,
   hchk, jmpcoms;                                                   034
   DECLARE EXTERNAL % not defined here see (nls,pdata,) %           054
      allsubs,                                                    03507
      nlssubs;                                                     055
   DECLARE COMMAND WORD                                           04279
      "BRANCH" = 1 ,                                              04825
      "GROUP" = 2 ,                                               04826
      "PLEX" = 3 ,                                                04827
      "STATEMENT" = 4 ,                                           04828
      "CHARACTER" = 5 ,                                           04829
      "CONTROLCHAR" = 6 ,                                         04830
      "INVISIBLE" = 7 ,                                           04831
      "LINK" = 8 ,                                                04832
      "DIRECTORY" = 9 ,                                           04833
      "PASSWORD" = 10 ,                                           04834
      "NUMBER" = 11 ,                                             04835
      "TEXT" = 12 ,                                               04836
      "MESSAGE" = 12 ,  %for Interrogate -- to get around CLI glitch%
                                                                  05202
      "VISIBLE" = 13 ,                                            04837
      "WORD" = 14 ,                                               04838
      "FILE" = 15 ,                                               04839
```

```
    "NEWFILELINK" = 16 ,                                                04840
    "OLDFILELINK" = 17 ,                                                04841
    "NAME" = 18 ,                                                       04842
    "IDENT" = 19 ,                                                      04843
    "IDENTLIST" = 20 ,                                                  04844
    "EDGE" = 21 ,                                                       04845
    "MARKER" = 22 ,                                                     04846
    "NLS" = 23 ,                                                        04847
    "ITEM" = 24 ,                                                       04848
    "ITEMNOVS" = 25 ,                                                   04849
    "SUCCESSOR" = 26 ,                                                  04850
    "PREDECESSOR" = 27 ,                                                04851
    "UP" = 28 ,                                                         04852
    "DOWN" = 29 ,                                                       04853
    "HEAD" = 30 ,                                                       04854
    "TAIL" = 31 ,                                                       04855
    "END" = 32 ,                                                        04856
    "BACK" = 33 ,                                                       04857
    "NEXT" = 34 ,                                                       04858
    "ORIGIN" = 35 ,                                                     04859
    "FILERETURN" = 36 ,                                                 04860
    "RETURN" = 37 ,                                                     04861
    "FILENAME" = 38 ,                                                   04862
    "FIRSTNAME" = 39 ,                                                  04863
    "NEXTNAME" = 40 ,                                                   04864
    "EXTNAME" = 41 ,                                                    04865
    "FIRSTCONTENT" = 42 ,                                               04866
    "NEXTCONTENT" = 43 ,                                                04867
    "FIRSTWORD" = 44 ,                                                  04868
    "NEXTWORD" = 45 ,                                                   04869
    "DETACHED" = 46 ,                                                   04870
    "TTY" = 47 ,                                                        04871
    "AUTO" = 48 ,                                                       04872
    "CONTINUE" = 49 ,                                                   04873
    "ON" = 50 ,                                                         04874
    "RECOVER" = 51 ,                                                    04875
    "SLINKER" = 52 ,                                                    04876
    "UPDATE" = 53 ,                                                     04877
    "CLEAR" = 54 ,                                                      04878
    "IDENTS" = 55 ,                                                     04879
    "FILES" = 56 ,                                                      04880
    "DELETE" = 57 ,                                                     04881
    "DEFERRED" = 58 ,                                                   04882
    "IMMEDIATE" = 59 ,                                                  04883
    "NOT" = 60 ,                                                        04884
    "PREVENT" = 61 ,                                                    04885
    "RESET" = 62 ,                                                      04886
    "ARCHIVE" = 63 ,                                                    04887
    "SEQUENTIAL" = 64 ,                                                 04888
    "TWO" = 65 ,                                                        04889
    "JUSTIFIED" = 66 ,                                                  04890
    "ASSEMBLER" = 67 ,                                                  04891
    "BOTH" = 68 ,                                                       04892
    "UNDELETE" = 69 ,                                                   04893
    "FOR" = 70 ,                                                        04894
    "STATUS" = 71 ,                                                     04895
```

```
    "TAPE" = 72 ,                                                   04896
    "ACCOUNT" = 73 ,                                                04897
    "NO" = 74 ,                                                     04898
    "VERSIONS" = 75 ,                                               04899
    "EXTENSION" = 76 ,                                              04900
    "DATE" = 77 ,                                                   04901
    "CREATION" = 78 ,                                               04902
    "LAST" = 79 ,                                                   04903
    "FIRST" = 80 ,                                                  04904
    "READ" = 81 ,                                                   04905
    "WRITE" = 82 ,                                                  04906
    "DUMP" = 83 ,                                                   04907
    "EVERYTHING" = 84 ,                                             04908
    "LENGTH" = 85 ,                                                 04909
    "MISCELLANEOUS" = 86 ,                                          04910
    "ACCESSES" = 87 ,                                               04911
    "PROTECT" = 88 ,                                                04912
    "SIZE" = 89 ,                                                   04913
    "TIME" = 90 ,                                                   04914
    "VERBOSE" = 91 ,                                                04915
    "SORT" = 92 ,                                                   04916
    "BYTESIZE" = 93 ,                                               04917
    "ARCHIVED" = 94 ,                                               04918
    "ALL" = 95 ,                                                    04919
    "MODIFICATIONS" = 96 ,                                          04920
    "UPPER" = 97 ,                                                  04921
    "LOWER" = 98 ,                                                  04922
    "MODE" = 99 ,                                                   04923
    "SENDMAIL" = 100 ,                                              04924
    "BUSY" = 101 ,                                                  04925
    "QUICKPRINT" = 102 ,                                            04926
    "JOURNAL" = 103 ,                                               04927
    "PRINTER" = 104 ,                                               04928
    "COM" = 105 ,                                                   04929
    "TERMINAL" = 106 ,                                              04930
    "REMOTE" = 107 ,                                                04931
    "REST" = 108 ,                                                  04932
    "CASE" = 109 ,                                                  04933
    "CONTENT" = 110 ,                                               04934
    "TEMPORARY" = 111 ,                                             04935
    "VIEWSPECS" = 112 ,                                             04936
    "EXTERNAL" = 113 ,                                              04937
    "TO" = 114 ,                                                    04938
    "PRIVATE" = 115 ,                                               04939
    "PUBLIC" = 116 ,                                                04940
    "TENEX" = 117 ,                                                 04941
    "ALLOW" = 118 ,                                                 04942
    "EXECUTE" = 119 ,                                               04943
    "APPEND" = 120 ,                                                04944
    "LIST" = 121 ,                                                  04945
    "SET" = 122 ,                                                   04946
    "SELF" = 123 ,                                                  04947
    "FORBID" = 124 ,                                                04948
    "DISK" = 125 ,                                                  04949
    "DEFAULT" = 126 ,                                               04950
    "OLD" = 127 ,                                                   04951
```

```
    "NEW" = 128 ,                                                   04952
    "COMPACT" = 129 ,                                               04953
    "RENAME" = 130 ,                                                04954
    "ADD" = 131 ,                                                   04955
    "SUBTRACT" = 132 ,                                              04956
    "MULTIPLY" = 133 ,                                              04957
    "DIVIDE" = 134 ,                                                04958
    "RIGHT" = 135 ,                                                 04959
    "LEFT" = 136 ,                                                  04960
    "ACTION" = 137 ,                                                04961
    "AUTHORS" = 138 ,                                               04962
    "COMMENT" = 139 ,                                               04963
    "EXPEDITE" = 140 ,                                              04964
    "HARDCOPY" = 141 ,                                              04965
    "INFORMATION" = 142 ,                                           04966
    "INSERT" = 143 ,                                                04967
    "KEYWORDS" = 144 ,                                              04968
    "OBSOLETES" = 145 ,                                             04969
    "RFC" = 146 ,                                                   04970
    "SUBCOLLECTIONS" = 147 ,                                        04971
    "TITLE" = 148 ,                                                 04972
    "UNRECORDED" = 149 ,                                            04973
    "L10" = 150 ,                                                   04974
    "PROCEDURE" = 151 ,                                             04975
    "SEQGEN" = 152 ,                                                04976
    "BUFFER" = 153 ,                                                04977
    "NDDT" = 154 ,                                                  04978
    "PARSERULE" = 155 ,                                             04979
    "CA" = 156 ,                                                    04980
    "CD" = 157 ,                                                    04981
    "RPT" = 158 ,                                                   04982
    "BC" = 159 ,                                                    04983
    "BW" = 160 ,                                                    04984
    "BS" = 161 ,                                                    04985
    "LITESC" = 162 ,                                                04986
    "IGNORE" = 163 ,                                                04987
    "SC" = 164 ,                                                    04988
    "SW" = 165 ,                                                    04989
    "TAB" = 166 ,                                                   04990
    "IMLAC" = 167 ,                                                 04991
    "TI" = 168 ,                                                    04992
    "NVT" = 169 ,                                                   04993
    "EXECUPORT" = 170 ,                                             04994
    "MENU" = 171 ,                                                  04995
    "DNLS" = 172 ,                                                  04996
    "TNLS" = 173 ,                                                  04997
    "COMMAND" = 174 ,                                               04998
    "RULE" = 175 ,                                                  04999
    "SUBSYSTEM" = 176 ,                                             05000
    "DISPLAY" = 177 ,                                               05001
    "FROZEN" = 178 ,                                                05002
    "HLPCOM" = 179 ,                                                05003
    "PROGRAM" = 180 ,                                               05004
    "TERSE" = 181 ,                                                 05005
    "INDENTING" = 182 ,                                             05006
    "UNIVERSAL" = 183 ,                                             05007
```

```
      "ENTRY" = 184 ,                                                    05008
      "INCLUDE" = 185 ,                                                  05009
      "BOTTOM" = 186 ,                                                   05010
      "PAGE" = 187 ,                                                     05011
      "OFF" = 188 ,                                                      05012
      "FULL" = 189 ,                                                     05013
      "PARTIAL" = 190 ,                                                  05014
      "ANTICIPATORY" = 191 ,                                             05015
      "DEMAND" = 192 ,                                                   05016
      "FIXED" = 193 ,                                                    05017
      "CONTROL" = 194 ,                                                  05018
      "CURRENTCONTEXT" = 195 ,                                           05019
      "FEEDBACK" = 196 ,                                                 05020
      "HERALD" = 197 ,                                                   05021
      "PRINTOPTIONS" = 198 ,                                             05022
      "PROMPT" = 199 ,                                                   05023
      "RECOGNITION" = 200 ,                                              05024
      "STARTUP" = 201 ,                                                  05025
      "LEVELADJUST" = 202 ,                                              05026
      "REVERSE" = 203 ,                                                  05027
      "TEST" = 204 ,                                                     05028
      "TASKER" = 205 ,                                                   05029
      "LINEPROCESSOR" = 206 ,                                            05030
      "CENTER" = 207 ,                                                   05031
      "CNTLQ" = 208 ,                                                    05032
      "OUTPUT" = 209 ,                                                   05041
      "SPACE" = 210 ,                                                    05081
      "WRAPAROUND" = 211,                                                05113
      "COMP80" = 212,                                                    05303
      "SINGER" = 213,                                                    05304
      "800VIDEOCOMP" = 214,                                              05305
      "500VIDEOCOMP" = 215;                                              05306
% COMMON RULES %                                                          03
   % ENTITY DEFINITIONS %                                                 04
      editentity = textent / structure;                                  05
   % TEXT ENTITY DEFINITIONS %                                           06
      textent = text1 / "TEXT" / "LINK" / "NUMBER";                      07
      text1 = "CHARACTER" / "WORD" / "VISIBLE" / "INVISIBLE";            08
   % STRUCTURE ENTITY DEFINITIONS %                                      09
      structure = "STATEMENT" / notstatement;                           010
      notstatement = "GROUP" / "BRANCH" / "PLEX" ;                      011
   % SWITCH %                                                           014
      switch = ("ON"! 1!/"OFF"!2 L2!);                                  015
   % DEVICE TYPES %                                                   01832
      devopt =                                                         01833
         ( "TASKER"!L2!                                                 0145
         / "TI" <"Terminal">                                            0146
         / "NVT"                                                       01839
         / "LINEPROCESSOR"                                             01834
         / "IMLAC"                                                     01835
         / "EXECUPORT"                                                  0147
         / "33-TTY"! 33!                                                0148
         / "35-TTY"! 35 L2!                                             0149
         / "37-TTY"! 37 L2! );                                          0150
% NLS EDITOR COMMANDS %                                                  056
    SUBSYSTEM nlseditor KEYWORD "BASE"                                   057
```

```
COMMAND %verify%                                                        058
   zverify =                                                            059
      "VERIFY" "FILE" CONFIRM xverify( ) ;                              060
COMMAND %update%                                                        061
   zupdate =                                                            062
      "UPDATE" "FILE"                                                   063
         namfil _ NULL                                                  064
         ent _ #"NEW"                                                   065
         (readconfirm()                                                01430
         / ent _                                                       01431
            ( "OLD" <"version">                                         067
            / "COMPACT"                                                 068
            / "RENAME" <"filename">                                     069
               namfil _ LSEL( #"NEWFILELINK" ) )                        070
            CONFIRM)                                                    071
         xupdate( ent, namfil ) ;                                       072
COMMAND %undelete%                                                      073
   zundelete =                                                          074
      "UNDELETE"!L2!                                                    075
         ent _                                                          076
            ( ( "FILE"                                                  077
               %/ "ARCHIVE" <"file">%                                   078
               ) namfil _ LSEL( #"OLDFILELINK")                        01845
            / "MODIFICATIONS" <"to file"> namfil _ NULL                 079
            )                                                           01846
         CONFIRM                                                        081
         xundelete( ent, namfil ) ;                                     082
COMMAND %trim%                                                          083
   ztrim =                                                              084
      "TRIM"!L2! "DIRECTORY"                                            085
         <"no. versions to keep"> param _ LSEL(#"NUMBER")               086
         CONFIRM <..."really?">                                        01497
         CONFIRM                                                        088
         xtrim( param ) ;                                               089
COMMAND %transpose%                                                     090
   ztranspose =                                                         091
      "TRANSPOSE"                                                       092
         sent _ editentity                                              093
         <"at"> source _ DSEL( sent )                                   094
         <"and"> dent _ sent                                            095
         dest _ DSEL( dent )                                            096
         vs _ NULL  filtre _ FALSE                                      097
            [filtre _ TRUE <"Filtered:"> vs _ VIEWSPECS]                098
         CONFIRM                                                        099
         xtranspose( sent, source, dent, dest, filtre, vs ) ;          0100
COMMAND %substitute%                                                   0101
   zsubstitute =                                                       03254
      "SUBSTITUTE"                                                     03255
         filtre _ FALSE  vs _ NULL  param _ NULL                       03256
         sent _ textent                                                03257
         <"in">                                                        03258
            ( readoption()                                             03259
               filtre _ TRUE <"Filtered:"> vs _ VIEWSPECS              03260
               dent _ structure                                        03261
            / dent _ structure )                                       03262
         <"at"> dest _ DSEL(dent)                                      03263
```

```
        % collect pairs of entities of type sent %                    03264
          subsinit( sent) % initialize collection stack %             03266
          PERFORM subst1 UNTIL                                        03267
             ( <"Finished?">                                          04256
                ( lookstat(sent) <"Finished?"> sublookansw()          04257
                / answ()  )                                           04258
             )                                                        04259
          CONFIRM                                                     03268
          xsubstitute( sent, dent, dest, param, filtre, vs );         03269
          subst1 =                                                    03270
             CLEAR                                                    03265
             sub1dsp(sent , TRUE)  %display <NEW etc > %              03271
             s1 _ LSEL(sent)                                          03272
             %sub2dsp( sent, s1 )  feedback param if bugged %         03273
             CLEAR                                                    05176
             sub1dsp(sent, FALSE)  %display <for old etc> %           03274
             s2 _ LSEL(sent)                                          03275
             %sub2dsp( sent, s2 )  feedback param if bugged %         03276
             param _ subpsave( sent, s1, s2) % stash the pairs away %;
                                                                      03277
COMMAND %stop%                                                        0124
   zstop =                                                            0125
      "STOP"!L2! "RECORD" <"of Session"> CONFIRM                      0126
      xstop() ;                                                       0127
COMMAND %sort%                                                        0134
   zsort =                                                            0135
      "SORT"!L2!                                                      0136
         dent _ notstatement                                          0137
         dest _ DSEL( dent )                                          0138
         CONFIRM                                                      0139
         xsort( dent, dest ) ;                                        0140
COMMAND %simulate%                                                    0141
   zsimulate =                                                        0142
      "SIMULATE"!L2! <"terminal type">                                0143
         ent _ devopt                                                 0144
         CONFIRM                                                      0151
         xsimulate( ent ) ;                                           0152
COMMAND %show%                                                        0153
   zshow =                                                            0154
      "SHOW"!L2!                                                      0155
      dent _ NULL                                                     01688
      filtre _ NULL                                                   01689
      param _ NULL                                                    01703
      ( "FILE"  ent _ #"FILE"                                         0157
         param _                                                      0158
            ( "STATUS"                                                0159
            / "DEFAULT" <"directory for links">                       0160
            / "MODIFICATIONS" <"status">                              0162
            / "RETURN" <"ring">                                       0163
            / "SIZE"!L2! )                                            0164
      / "RETURN" <"ring">                                             03535
         ent _ #"RETURN"                                              03536
      / "MARKER" <"list">                                             0161
         param _ #"MARKER" ent _ #"FILE"                              03315
      %/ "ARCHIVE" <"directory">  ent _ #"ARCHIVE"                    0165
         param _ (readconfirm() / LSEL(#"NAME") )%                    0166
```

```
          %CAN WE DO THIS???%                                          0167
      / "DIRECTORY"  ent _ #"DIRECTORY"                                01690
        <"of">                                                         01691
          ( readconfirm()                                              01704
          / param _  LSEL(#"DIRECTORY") CONFIRM                        01705
          )                                                            01706
        dent _ seqbinit()                                              01692
        [optdirs]                                                      05170
      / "DISK"!L2! <"space status">  ent _ #"DISK"                     0171
        param _ NULL                                                   0172
      / "NAME" <"delimiters for statement at">  ent _ #"NAME"          0173
        param _ DSEL( #"STATEMENT" )                                   0174
      / "VIEWSPECS" <"status">  ent _ #"VIEWSPECS"                     0175
        param _ NULL                                                   0176
        [param _ "VERBOSE"] )                                          0177
      CONFIRM                                                          0178
      xshow( ent, param, dent ) ;                                      0179
      optdirs = CLEAR <"opt:"> PERFORM diropt UNTIL (<"Finished?">
      lookansw());                                                     05171
COMMAND %set%                                                          0180
   zset =                                                              0181
      "SET"!L2!                                                        0182
      dest _ NULL param _ NULL  param2 _ NULL param3 _ NULL            0183
      (                                                                0184
        % ent _ "CHARACTER"!NOTT L2!                                   0191
          <"size for window to"> param _ LSEL( #"NUMBER" )             0192
        /% ent _ "EXTERNAL" <"Names Link File To:">                    03171
          param _ LSEL(#"LINK")                                        03172
        / ent _ "CONTENT" <"pattern">                                  0195
          param _                                                      0196
            ( "TO"                                                     0197
              param2 _ LSEL( #"CHARACTER" )                            0198
            / switch )                                                 0199
        / ent _ "LINK"                                                 0200
          <"default for file to directory">                            0201
          param2 _ LSEL(#"NAME")                                       0202
        / ent _ "NAME"                                                 0203
          <"delimiters in"> param _ structure                          0204
          <"at"> dest _ DSEL( param )                                  0205
          CLEAR                                                        0206
          <"left delimiter"> param2 _ LSEL(#"CHARACTER")               0207
          <"right delimiter"> param3 _ LSEL(#"CHARACTER")              0208
        / "NLS"!L2! <"protection for file">                            03781
          ent _ ( "PRIVATE" / "PUBLIC"!L2! )                           03064
        / ent _ "TENEX" <"protection for file named">                  03780
          param _ LSEL(#"OLDFILELINK")                                 01714
            param2 _ seqbinit()                                        01715
            ( "SET" <"to">                                             01716
              param3 _ #"SET"                                          02201
              seqbadd( param2, param3)                                 02202
              param3 _ LSEL(#"NUMBER")                                 01717
              seqbadd( param2, param3)                                 01718
            / "ALLOW"                                                  01719
              param3 _ #"ALLOW"                                        01720
              seqbadd( param2, param3)                                 01721
              prtgrp                                                   02199
```

```
                PERFORM prot1 UNTIL (<"Finished?"> lookansw())
                                                                    01723
              / "FORBID"                                            01724
                 param3 _ #"FORBID"                                 01725
                 seqbadd( param2, param3)                           01726
                 prtgrp                                             02200
                 PERFORM prot1 UNTIL (<"Finished?"> lookansw())
                                                                    01728
              / "RESET"                                             01729
                 param3 _ #"RESET"                                  01730
                 seqbadd( param2, param3)                           01731
              / "PRIVATE"                                           01732
                 param3 _ #"PRIVATE"                                01733
                 seqbadd( param2, param3)                           01734
                 <"for"> prtgrp                                     01735
              )                                                     01736
        / ent _ "TEMPORARY"!L2! <"modifications for file">           0213
           CONFIRM <"really?">                                      03782
        / ent _ "TTY"!NOTT L2! <"simulation for window">             0214
           lookbug() %for correct prompting%                        04146
           param _ DSEL(#"EDGE")                                     0215
        / ent _ "VIEWSPECS"                                          0216
           param _ VIEWSPECS )                                       0217
     CONFIRM                                                         0218
     xset( ent, param, param2, param3, dest ) ;                      0219
  prtgrp =                                                          03786
     fromwhom _                                                     03787
        ( "SELF"                                                    03788
        / "GROUP"                                                   03789
        / "PUBLIC"                                                  03790
        )                                                           03791
     seqbadd( param2, fromwhom );                                   03792
  prot1 =                                                           03793
     ( param3 _ "SET" <"to">                                        03794
        seqbadd( param2, param3 )                                   03795
        param3 _ LSEL(#"NUMBER")                                    03796
        seqbadd( param2, param3 )                                   03797
     /                                                              03798
        ( param3 _                                                  03799
           ( "READ"                                                 03800
           / "WRITE"                                                03801
           / "EXECUTE"                                              03802
           / "APPEND"!L2!                                           03803
           / "LIST"                                                 03804
           / "ALL"                                                  03805
           ) <"access"> seqbadd( param2, param3)                    03806
        )                                                           03807
     );                                                             03808
%COMMAND retrieve%                                                   0220
  %zretrieve =                                                       0221
     "RETRIEVE"!L2! "FILE" <"from archive">                          0222
        namfil _ LSEL(#"OLDFILELINK")                                0223
        CONFIRM                                                      0224
        xretrieve( namfil );%                                        0225
COMMAND %reset%                                                      0226
  zreset =                                                           0227
```

```
      "RESET"!L2!                                                          0228
         dest _ NULL dent _ NULL                                           0229
         ent _                                                             0230
            ( "ARCHIVE" <"request for file">                               0231
               dest _ LSEL(#"OLDFILELINK")                                 0232
            / "CASE" <"mode">                                              0233
            %/ "CHARACTER"!NOTT L2! <"size for window">%                   0234
            / "CONTENT"!L2! <"Pattern">                                    0236
            / "LINK" <"default for file">                                  0237
            / "NAME" <"delimiters in">                                     0238
               dent _ structure                                            0239
               <"at"> dest _ DSEL(dent)                                    0240
            / "TEMPORARY" <"modifications for file">                       0243
            / "TTY"!NOTT L2! <"window">                                    0244
            / "VIEWSPECS" )                                                0245
         CONFIRM                                                           0246
         xreset( ent, dent, dest );                                        0247
COMMAND %replace%                                                          0248
   zreplace =                                                              0249
      "REPLACE"                                                            0250
         dent _ editentity                                                 0251
         <"at"> dest _ DSEL(dent)                                          0252
         sent _ dent                                                       0253
         <"by"> source _ LSEL(sent)                                        0254
         CONFIRM                                                           0255
         xreplace( dent, dest, sent, source );                             0256
COMMAND %renumber%                                                         0257
   zrenumber =                                                             0258
      "RENUMBER"!L2! "SIDS"                                                0259
         <"in file"> CONFIRM                                               0260
         xrenumber( );                                                     0261
COMMAND %release%                                                          0262
   zrelease =                                                              0263
      "RELEASE"!NOTT L2!                                                   0264
         ent _                                                             0265
            ( "FROZEN" <"statement at">                                    0266
               dest _ DSEL(#"STATEMENT")                                   0267
            / "ALL" <"frozen statements"> dest _ NULL )                    0268
         CONFIRM                                                           0269
         xrelease( ent, dest );                                            0270
COMMAND %start%                                                           05034
   zstart =                                                                05035
      "START"!L2! "RECORD" <"of Session on file"> namfil _
      LSEL(#"NEWFILELINK")                                                 05036
         ent _ FALSE                                                       05037
         ( readconfirm()                                                   05038
         / "RUNFILE" <"format"> ent _ TRUE CONFIRM )                       05039
         xstart( namfil, ent );                                            05040
COMMAND %process dnls%                                                    01513
   zdprocess =                                                             01514
      "PROCESS" ! NOTT! process;                                           01515
   process =                                                               04227
      <"Commands from"> dent _ structure                                   01524
      <"at"> dest _ DSEL(dent)                                             01523
      CONFIRM                                                              01525
      xprocess( dent, dest );                                              01520
```

```
COMMAND %process tnls%                                                  04220
   ztprocess =                                                          04221
      "PROCESS" !NOTD L2! process;                                      04222
COMMAND %print%                                                          0299
   zprint =                                                              0300
      "PRINT"! NOTD!                                                     0301
         ( ent _ "REST" CONFIRM  dest _ NULL vs _ NULL                   0302
         / ent _ "FILE" CONFIRM  dest _ NULL vs _ NULL                  03534
         / ent _ structure                                               0303
            <"at"> dest _ DSEL( ent )                                    0304
            vs _ VIEWSPECS                                               0305
         / ent _ "JOURNAL" <"mail"> CONFIRM dest _ NULL vs _ NULL)
                                                                        03150
         xprint( ent, dest, vs );                                        0306
COMMAND %playback%                                                       0307
   zplayback =                                                           0308
      "PLAYBACK"!L2! "RECORD"                                            0309
         <"of Session from file"> namfil _ LSEL(#"OLDFILELINK")          0310
         CLEAR                                                          04253
         <"Simulate Recorded timing?"> param2 _ answer()                04252
         CONFIRM                                                         0311
         xplayback( namfil , param2  );                                  0312
COMMAND %output%                                                        05212
   zoutput =                                                            05213
      "OUTPUT" <"to"> xoutsapf( FALSE )  xoutsnhf( FALSE )              05214
         (                                                              05215
            (                                                           05216
               ( ent _ "JOURNAL"  <"Quickprint"> param2 _ FALSE
                                                                        05217
               / ent _ "PRINTER" param2 _ FALSE                         05218
               / ent _ "COM" comexflag _ #"COMP80" <"for device">
                                                                        05219
                  (lookconfirm()                                        05220
                  / comexflag _ ("SINGER"                               05221
                     /"COMP80"                                          05222
                     /"500VIDEOCOMP"                                    05307
                     /"800VIDEOCOMP")                                   05223
                  )                                                     05224
               )                                                        05225
               namfil _ NULL                                            05226
               param _ TRUE % use default number of copies %            05227
               param2 _ FALSE % Not test. %                             05228
               ( lookconfirm()                                          05229
               / "APPEND" <"to File">                                   05230
                  namfil _ LSEL(#"OLDFILELINK") xoutsapf( TRUE )
                                                                        05231
               / "FILE" namfil _ LSEL(#"NEWFILELINK")                   05232
               / "COPIES" param _ LSEL(#"NUMBER")                       05233
               / "TEST" param2 _ #"TEST"                                05234
                  ( lookconfirm()                                       05235
                  / "FILE" namfil _ LSEL(#"NEWFILELINK")                05236
                  )                                                     05237
               )                                                        05238
               CONFIRM                                                  05239
               xout1( ent, namfil, param, param2 )                      05240
            )                                                           05241
```

```
/ ( ent _                                                          05242
   "QUICKPRINT"                                                    05243
      namfil _ NULL                                                05244
      param _ TRUE % use default number of copies %                05245
      param2 _ FALSE % Not test. %                                 05246
      ( lookconfirm()                                              05247
      / "NO" <"Headers"> xoutsnhf( TRUE )                          05248
         ( lookconfirm()                                           05249
         / "APPEND" <"to File">                                    05250
            namfil _ LSEL(#"OLDFILELINK") xoutsapf( TRUE )
                                                                   05251
         / "FILE" namfil _ LSEL(#"NEWFILELINK")                    05252
         / "COPIES" param _ LSEL(#"NUMBER")                        05253
         / "TEST" param2 _ #"TEST"                                 05254
         )                                                         05255
      / "APPEND" <"to File">                                       05256
         namfil _ LSEL(#"OLDFILELINK") xoutsapf( TRUE )
                                                                   05257
      / "FILE" namfil _ LSEL(#"NEWFILELINK")                       05258
      / "COPIES" param _ LSEL(#"NUMBER")                           05259
      / "TEST" param2 _ #"TEST"                                    05260
      )                                                            05261
      CONFIRM                                                      05262
      xout1( ent, namfil, param, param2 ))                         05263
   param _ TRUE % use default number of copies %                   05264
/ ( ent _                                                          05265
   ("SEQUENTIAL" /                                                 05266
   "ASSEMBLER" )                                                   05267
      ( "APPEND" <"to File"> xoutsapf(TRUE)                        05268
         namfil _ LSEL(#"OLDFILELINK")                             05269
      / "FILE"                                                     05270
         namfil _ LSEL(#"NEWFILELINK")                             05271
      )                                                            05272
      param _ FALSE                                                05273
         ( lookconfirm()                                           05274
         / "FORCE" <"upper case"> param _ TRUE                     05275
         )                                                         05276
      param2 _ FALSE                                               05277
      CONFIRM                                                      05278
      xout1( ent, namfil, param, param2 ))                         05279
/ (                                                                05280
   ( ent _ "TERMINAL" param _ NULL  port _ NULL                    05281
      ( lookconfirm()                                              05282
      / "FILE" param _ LSEL(#"NEWFILELINK")                        05283
      )                                                            05284
   / ent _ "REMOTE"                                                05285
      <"printer -- TIP"> param _ LSEL(#"VISIBLE")                  05286
      <"Port #"> port _ LSEL(#"NUMBER") )                          05287
   CONFIRM                                                         05288
   CLEAR                                                           05289
   <"Send Form Feeds?">                                            05290
      (answ() ff _ TRUE  sim _ FALSE                               05291
         / <"Simulate?"> ff _ FALSE sim _ answer())                05292
   CLEAR                                                           05293
   <"Wait at page break?"> pb _ answer()                           05294
   CLEAR                                                           05295
```

```
            <"Go?">                                                    05296
               ( answ()                                                05297
               /                                                       05298
                  <"Type CA when ready, CD to abort">                  05299
                  readconfirm())                                       05300
            xout2( ent, param, port, ff, sim, pb))                     05301
         );                                                            05302
COMMAND %move%                                                          0360
   zmove =                                                              0361
      "MOVE"                                                            0362
         filtre _ FALSE                                                 0363
         vs _ NULL                                                      0364
         level _ NULL                                                   0365
         ( sent _ "LINK"                                                0366
            copy1                                                      02952
            dent _ sent                                                02953
            dest _ DSEL(#"VISIBLE")                                    02954
         / sent _ "NUMBER"                                             04237
            copy1                                                      04238
            dent _ sent                                                04239
            dest _ DSEL(#"VISIBLE")                                    04240
         / sent _ text1                                                02951
            copy1                                                       0367
            dent _ sent                                                 0368
            dest _ DSEL(dent)                                           0369
         / sent _ "TEXT"                                                0370
            copy1                                                       0371
            dent _ #"CHARACTER"                                         0372
            dest _ DSEL(dent)                                           0373
         / sent _ structure                                             0374
            copy1                                                       0375
            dent _ #"STATEMENT"                                         0376
            dest _ DSEL(dent)                                           0377
            [filtre _ TRUE <"Filtered:"> vs _ VIEWSPECS]                0378
            level _ LEVADJ                                              0379
         / dent _ "FILE"                                                0380
            sent _ dent                                                 0381
            <"from old filename"> source _ LSEL(#"OLDFILELINK")         0382
            <"to new filename"> dest _ LSEL(#"NEWFILELINK")             0383
         / sent _ "EDGE"! NOTT!                                         0384
            <"from"> source _ DSEL(#"EDGE")                             0386
            <"to">                                                     03638
               (lookbug() dent _ NULL / dent _ "CENTER" <"of">)
                                                                       03425
               dest _ DSEL(#"EDGE")                                    03426
         )                                                              0388
         CONFIRM                                                        0389
         xmove(sent, source, dent, dest, level, filtre, vs);            0390
%COMMAND %%merge%%                                                      0391
   zmerge =                                                             0392
      "MERGE"!L2!                                                       0393
         sent _ notstatement                                            0394
         <"at"> source _ SSEL(sent)                                     0395
         dent _ sent                                                    0396
         <"into"> dest _ DSEL(dent)                                     0397
         CONFIRM                                                        0398
```

```
         xmerge( sent, source, dent, dest);%                          0399
COMMAND %mark%                                                        0400
   zmark =                                                            0401
      "MARK"!L2! "CHARACTER" <"at">                                   0402
         dest _ DSEL( #"CHARACTER" )                                  0403
         <"with marker named"> source _ LSEL(#"NAME")                 0404
         CONFIRM                                                      0405
         xmark( dest, source);                                        0406
COMMAND %logout%                                                      0407
   zlogout =                                                          0408
      "LOGOUT"!L2!                                                    0409
         CONFIRM xlogout();                                           0410
COMMAND %load%                                                        0411
   zload =                                                            0412
      "LOAD"                                                          0413
         ent _ "FILE"                                                 0414
         namfil _ LSEL(#"OLDFILELINK") CONFIRM                        0415
         xload(ent, namfil);                                          0416
%k (unused)%                                                          0417
COMMAND %insert%                                                      0499
   zinsert =                                                          0500
      "INSERT"                                                        0501
         level _ NULL                                                 0502
         ( ent _ "LINK"                                               0503
            <"to follow">                                             0504
            dest _ DSEL(#"VISIBLE")                                   0505
            param _ LSEL(ent)                                         0506
         / ent _ "NUMBER"                                            04241
            <"to follow">                                             04242
            dest _ DSEL(#"VISIBLE")                                   04243
            param _ LSEL(ent)                                         04244
         / ent _ text1                                                02959
            <"to follow">                                             02960
            dest _ DSEL(ent)                                          02961
            param _ LSEL(ent)                                         02962
         / ent _ "TEXT"                                               0507
            <"to follow">                                             0508
            dest _ DSEL(#"CHARACTER")                                 0509
            param _ LSEL(ent)                                         0510
         / ent _ structure                                            0511
            <"to follow">                                             0512
            dest _ DSEL(#"STATEMENT")                                 0513
            level _ LEVADJ                                            0514
            param _ LSEL(ent)                                         0515
         / ent _ "DATE"                                               02191
            <"to follow">                                             02192
            dest _ DSEL(#"VISIBLE")                                   02193
            param _ NULL                                              02194
         / ent _ "TIME"!L2!                                           02195
            <"and Date to follow">                                    02196
            dest _ DSEL(#"VISIBLE")                                   02197
            param _ NULL                                              02198
         / ent _ "SENDMAIL"!L2! <"form">                              0516
            <"to follow">                                             0517
            dest _ DSEL(#"STATEMENT")                                 0518
            level _ LEVADJ                                            0519
```

```
            param _ NULL                                                0520
         / ent _ "EDGE"! NOTT! <"perpendicular to">                    03382
            ( lookbug() param _ NULL                                    03383
            / param _ "CENTER" <"of"> )                                 03423
            dest _ DSEL(#"EDGE")                                        03424
         )                                                              0521
         CONFIRM                                                        0522
         xinsert(ent, dest, level, param);                              0523
   zinsstatement = % INS STATEMENT FUNCTION %                           0524
      CLEAR                                                             02913
      level _ LEVADJ                                                    0525
      param _ LSEL(#"STATEMENT")                                        0526
      CONFIRM xinsstatement(level, param );                             0527
% h (unused) %                                                         0528
COMMAND %freeze%                                                       0529
   zfreeze =                                                           0530
      "FREEZE"!NOTT L2! "STATEMENT" <"at">                             0531
         dest _ DSEL(#"STATEMENT")                                     0532
         vs _ VIEWSPECS CONFIRM                                        0533
         xfreeze(dest, vs);                                            0534
% COMMAND format%                                                     02212
COMMAND %force case%                                                   03309
   zforce =                                                            03310
      "FORCE" <"Case">                                                 03311
         ( param _ editentity                                          03319
            <"at"> dest _ DSEL( param )                                03320
            param2 _ NULL  [CLEAR <"opt:"> cshmode]                    03321
         / param _ "MODE" dest _ NULL                                  03322
            cshmode                                                    03323
         )                                                             03324
         CONFIRM                                                       03325
         xforce( param, param2, dest ) ;                               03326
   % CASE SHIFT MODES %                                                016
      cshmode = param2 _                                               017
         ( "UPPER"                                                     018
         / "LOWER"                                                     019
         / "FIRST" <"letter upper"> );                                 020
COMMAND %expunge%                                                      0535
   zexpunge =                                                          0536
      "EXPUNGE"!L2! <"deleted files from">                             0537
         ent _                                                         0538
            ( "DIRECTORY"                                              0539
            %/ "ARCHIVE" <"directory">% )                              0540
         CONFIRM                                                       0541
         xexpunge(ent);                                                0542
COMMAND %edit%                                                         0543
   zedit =                                                             0544
      "EDIT"!NOTD L2! "STATEMENT" <"at">                               0545
         dest _ DSEL(#"STATEMENT")                                     0546
         xedit(dest);                                                  0547
COMMAND %disconnect%                                                   0548
   zdisconnect =                                                       0549
      "DISCONNECT"!L2! "TERMINAL" CONFIRM                              0550
      xdisconnect();                                                   0551
COMMAND %delete%                                                       0552
   zdelete =                                                           03278
```

```
      "DELETE"                                                          03279
         filtre _ FALSE dest _ NULL vs _ NULL                           03280
         (                                                              03281
            ent _ textent                                               03282
               <"at"> dest _ DSEL(ent)                                  03283
               needconfirm _ TRUE % to force confirmation %             03284
            / ent _ structure                                           03285
               <"at"> dest _ DSEL(ent)                                  03286
               [                                                        03287
                  filtre _ TRUE <"Filtered:">                           03288
                  vs _ VIEWSPECS                                        03289
                  needconfirm _ TRUE % to force confirmation %          03290
               ]                                                        03291
            / ent _ ("FILE" %/ "ARCHIVE"!L2! <"file">%)                 03292
               dest _ LSEL(#"OLDFILELINK")                              03293
               needconfirm _ TRUE % to force confirmation %             03294
            / "MARKER"!L2! <"named">                                    03295
               ent _ #"MARKER"                                          03296
               dest _ LSEL(ent)                                         03297
               needconfirm _ TRUE % to force confirmation %             03298
            / "EDGE"! NOTT! <"at">                                      03427
               ent _ #"EDGE"                                            03428
               dest _ DSEL(ent)                                         03429
            / "ALL" <"markers">                                         03299
               ent _ #"ALL"                                             03300
               needconfirm _ TRUE % to force confirmation %             03301
            / "MODIFICATIONS" <"to file">                               03302
               ent _ #"MODIFICATIONS"                                   03303
               CONFIRM <"really?">                                      03304
               needconfirm _ TRUE % to force confirmation %             03305
         )                                                              03306
         CONFIRM                                                        03307
         xdelete(ent, dest, filtre, vs);                                03308
COMMAND %create%                                                         0573
   zcreate =                                                             0574
      "CREATE"!L2! "FILE"                                                0575
         namfil _ LSEL(#"NEWFILELINK") CONFIRM                           0576
         xcreate(namfil);                                                0577
COMMAND %copy%                                                           0578
   zcopy =                                                               0579
      "COPY"                                                             0580
         vs _ NULL                                                       0581
         level _ NULL                                                    0582
         source _ NULL                                                  01707
         filtre _ NULL                                                   0583
         dest _ NULL                                                     0584
         ( sent _ "LINK"                                                 0585
            copy1                                                        0586
            dent _ sent                                                  0587
            dest _ DSEL(#"VISIBLE")                                      0588
         / sent _ "NUMBER"                                              04245
            copy1                                                       04246
            dent _ sent                                                 04247
            dest _ DSEL(#"VISIBLE")                                     04248
         / sent _ text1                                                 02955
            copy1                                                       02956
```

```
            dent _ sent                                                 02957
            dest _ DSEL(dent)                                           02958
         / sent _ "TEXT"                                                 0589
            copy1                                                        0590
            dent _ #"CHARACTER"                                          0591
            dest _ DSEL(dent)                                            0592
         / sent _ structure                                              0593
            copy1                                                        0594
            dent _ #"STATEMENT"                                          0595
            dest _ DSEL(dent)                                            0596
            [filtre _ TRUE <"Filtered:"> vs _ VIEWSPECS]                 0597
            level _ LEVADJ                                               0598
         / sent _ "FILE"                                                 0599
            dent _ sent                                                  0600
            <"from"> source _ LSEL(#"OLDFILELINK")                       0601
            <"to a new file to be named"> dest _ LSEL(#"NEWFILELINK")
                                                                         0602
         / sent _ "DIRECTORY"                                            0603
            copy3 <"directory options:">                                 0604
            ( readoption() CLEAR <"opt:">                                0605
               PERFORM diropt UNTIL( <"Finished?"> lookansw())           0606
            / DUMMY) % dummy case %                                      0607
         %/ sent _ "ARCHIVE" <"directory">                               0608
            copy3 <"directory options:">                                 0609
            ( readoption()                                               0610
               PERFORM copy4 UNTIL( <"Finished?"> lookansw())            0611
            / DUMMY)% % dummy case %                                     0612
         / sent _ "SEQUENTIAL"!L2!                                      04264
            <"file from"> source _ LSEL(#"OLDFILELINK")                 04265
            <" to follow"> dest _ DSEL(#"STATEMENT")                    04266
            level _ LEVADJ                                              04267
            CLEAR <"using">                                             04269
            seqtype                                                     05178
         )                                                               0619
         CONFIRM                                                         0620
         xcopy(sent, source, dent, dest, level, filtre, vs);             0621
   copy1 =                                                               0622
      <"from"> source _ SSEL(sent)                                       0623
      <"to follow">;                                                     0624
   copy3 =                                                               0627
      <"of">                                                             0628
         ( readconfirm()                                                01709
         / source _ LSEL( #"DIRECTORY" )                                01710
         )                                                              01711
      <"to follow"> dest _ DSEL(#"STATEMENT")                            0629
      level _ LEVADJ                                                     0630
      dent _ seqbinit();                                                 0631
   copy4 = param2 _ diropt  seqbadd( dent, param2 );                     0633
   seqtype =                                                           05177
      ("ONE"  <"<CR> to end statement">                                04270
         dent _ NULL                                                   04268
      / "TWO"  <"<CR>s end statement">                                 04271
         (lookconfirm() dent _ #"TWO" /                                04272
         "JUSTIFIED"  <"delete extra <SP>">                            04273
            dent _ #"JUSTIFIED"                                        04277
         )                                                             04276
```

```
      / "ASSEMBLER" dent _ #"ASSEMBLER"                                  04274
      );                                                                 04278
COMMAND %connect%                                                         0634
   zconnect =                                                             0635
      "CONNECT"!L2!                                                       0636
         <"to">                                                           0637
         (ent _ ("DISPLAY"!L2! / "TTY")                                   0638
            <"Number"> dest _ LSEL(#"NUMBER") <"for">                     0639
            param _ ("INPUT"! 1! <"and Output"> / "OUTPUT" <"Only">)
            /                                                             0640
         ent _ "DIRECTORY"  dest _ LSEL(#"DIRECTORY")                     01697
            param _ NULL <"Password">                                     01698
            ( lookconfirm()                                               01699
            / param _ LSEL(#"PASSWORD")  )                                01700
            )                                                             01701
         CONFIRM                                                          0645
         xconnect(ent, dest, param);                                      0646
COMMAND %clear%                                                           0647
   zclear =                                                               0648
      "CLEAR"!NOTT L2! <"TTY Window"> CONFIRM xclear();                   0649
COMMAND %break%                                                           0652
   zbreak =                                                               0653
      "BREAK"                                                             0654
         ent _ "STATEMENT"                                                0655
            <"at"> dest _ DSEL(#"CHARACTER")                              0656
            level _ LEVADJ                                                0657
         CONFIRM                                                          0658
         xbreak(ent, dest, level); %should also pass literal%             0659
COMMAND %archive%                                                         0660
   zarchive =                                                             03189
      "ARCHIVE"!L2!                                                       03190
         ( "FILE"                                                         03191
            namfil _ LSEL(#"OLDFILELINK")                                 03192
            param _ segbinit()                                            03193
            [ arch1 ]                                                     03194
         ) CONFIRM xarchive(namfil, param);                               03195
         arch1 =                                                          03196
            CLEAR <"opt:">                                                03197
            PERFORM archopts UNTIL (lookansw());                          03198
         archopts =                                                       03199
            param2 _                                                      03200
               ( "DELETE" <"After Archiving">                             03201
               / "DEFERRED"!L2!                                           03202
               %   NOT IMPLEMENTED                                        03526
               / "IMMEDIATE"                                              03203
               %                                                          03527
               / "NOT" <"Allowed">                                        03204
               / "PREVENT" <"Deletion After Archiving">                   03205
               / "RESET" <"Request Status">                               03206
               )                                                          03207
            CONFIRM segbadd( param, param2 ) <"Finished?">;               03208
COMMAND %append%                                                          0672
   zappend =                                                              0673
      "APPEND"                                                            0674
      sent _ "STATEMENT"                                                  03528
      zapp1                                                               03529
```

```
      dent _ #"STATEMENT"                                                03530
      %      REST NOT IMPLEMENTED   (dash replaces percent)              03531
      ( sent _ "LINK"                                                     0675
         zapp1                                                            0676
         dent _ #"VISIBLE"                                                0677
      / sent _ "NUMBER"                                                   04249
         zapp1                                                            04250
         dent _ #"VISIBLE"                                                04251
      / sent _ text1 - text entities except "TEXT"!L2! -                  02963
         zapp1                                                            02964
         dent _ sent                                                      02965
      / sent _ "TEXT"                                                     0678
         zapp1                                                            0679
         dent _ #"CHARACTER"                                              0680
      / sent _ structure                                                  0681
         zapp1                                                            0682
         dent _ #"STATEMENT"                                              0683
      )                                                                   0684
      %                                                                   03532
      dest _ DSEL(dent)                                                   0685
      <"join with"> literal _ LSEL(#"TEXT")                               0686
      CONFIRM                                                             0687
      xappend(sent, source, dent, dest, literal);                         0688
   zapp1 =                                                                0689
      <"at">                                                              0690
      source _ SSEL(sent)                                                 0691
      <"to">;                                                             0692
COMMAND %accept%                                                          0693
   zaccept =                                                              0694
      "ACCEPT"!L2! "CONNECT"                                              0695
         <"from terminal number">  dest _ LSEL(#"NUMBER") <"for">
                                                                          0696
         param _ ("INPUT"! 1! <"and Output"> / "OUTPUT" <"Only">)
                                                                          03416
         CONFIRM xaccept( param, dest );                                  0697
COMMAND %period%                                                          0702
   zperiod =                                                              0703
      "."! NOTD!                                                          0704
         xperiod() ;                                                      0705
COMMAND %tab%                                                             0706
   ztab =                                                                 0707
      "      "                                                           0708
         xtab() ;                                                         0709
COMMAND %slash%                                                           0710
   zslash =                                                               0711
      "/"! NOTD!                                                          0712
         xslash() ;                                                       0713
COMMAND %bslash%                                                          0714
   zbslash =                                                              0715
      "\"! NOTD!                                                          0716
         xbslash() ;                                                      0717
COMMAND %uparrow%                                                         0718
   zuparrow =                                                             0719
      "^"! NOTD!                                                          0720
         xuparrow() ;                                                     0721
COMMAND %linefeed%                                                        0722
```

```
   zlinefeed =                                                          0723
      "<LF>"! NOTD!                                                     0724
         xlinefeed() ;                                                  0725
% DIRECTORY OPTIONS %                                                  01536
   % dummy rule to get dent defined properly %                         01537
      diropl = dent _ FALSE;                                            01538
   % options for what is to be listed %                                02184
   diropt =                                                             01539
      gparam _ FALSE                                                    01540
      gparm2 _ FALSE                                                    01541
      gparm3 _ FALSE                                                    01542
      gparm4 _ FALSE                                                    01543
                                                                        01544
      ( "ALL"!L2! <"Files">                                             01545
         gparam _ #"BOTH"                                               01546
      / "DELETE"!L2! <"Files Only">                                     01547
         gparam _ #"DELETE"                                             01548
      / "UNDELETE" <"Files Only">                                       01549
         gparam _ #"UNDELETE"                                           01550
      / "FOR" <"File">                                                  01551
         gparam _ #"FOR"                                                01553
         gparm2 _ LSEL(#"OLDFILELINK")                                  01552
      / "ARCHIVE"                                                       01554
         gparam _ #"ARCHIVE"                                            01555
         ( "STATUS"                                                     01556
            gparm2 _ #"STATUS"                                          01557
         / "TAPE" <"Numbers">                                           01558
            gparm2 _ #"TAPE"                                            01559
         )                                                              01560
      / "ACCOUNT"!L2!                                                   01561
         gparam _ #"ACCOUNT"                                            01562
      / "DATE" <"of">                                                   01563
         gparam _ #"DATE"                                               01564
         ( "ARCHIVE"                                                    01565
            gparm2 _ #"ARCHIVE"                                         01566
         / "CREATION"                                                   01567
            gparm2 _ #"CREATION"                                        01568
         / "LAST" <"Dump">                                              01569
            gparm2 _ #"LAST"                                            01570
         / "FIRST" <"Version Creation">                                 01571
            gparm2 _ #"FIRST"                                           01572
         / "READ"                                                       01573
            gparm2 _ #"READ"                                            01574
         / "WRITE"                                                      01575
            gparm2 _ #"WRITE"                                           01576
         )                                                              01577
      / "DUMP"!L2! <"Tape Number">                                      01578
         gparam _ #"DUMP"                                               01579
      / "EVERYTHING"                                                    01580
         gparam _ #"EVERYTHING"                                         01581
      / "LAST" <"Writer">                                               01582
         gparam _ #"LAST"                                               01583
      / "LENGTH"!L2! <"and Bytesize">                                   01584
         gparam _ #"LENGTH"                                             01585
      / "MISCELLANEOUS" <"Information">                                 01586
         gparam _ #"MISCELLANEOUS"                                      01587
```

```
      / "NUMBER" <"of">                                                 01588
        gparam _ #"NUMBER"                                              01589
        ( "VERSIONS" <"to keep">                                        01590
           gparm2 _ #"VERSIONS"                                         01591
        / "ACCESSES"                                                    01592
           gparm2 _ #"ACCESSES"                                         01593
        )                                                               01594
      / "NO"!L2!                                                        02431
        gparam _ #"NO"                                                  02432
        ( "VERSIONS" <"number">                                         02433
           gparm2 _ #"VERSIONS"                                         02434
        / "EXTENSION" <"name">                                          02435
           gparm2 _ #"EXTENSION"                                        02436
        )                                                               02437
      / "PROTECT"                                                       01595
        gparam _ #"PROTECT"                                             01596
      / "SIZE" <"in Pages">                                             01597
        gparam _ #"SIZE"                                                01598
      / "TIME" <"and Date of">                                          01599
        gparam _ #"TIME"                                                01600
        ( "ARCHIVE"                                                     01601
           gparm2 _ #"ARCHIVE"                                          01602
        / "CREATION"                                                    01603
           gparm2 _ #"CREATION"                                         01604
        / "LAST" <"Dump">                                               01605
           gparm2 _ #"LAST"                                             01606
        / "FIRST" <"Version Creation">                                  01607
           gparm2 _ #"FIRST"                                            01608
        / "READ"                                                        01609
           gparm2 _ #"READ"                                             01610
        / "WRITE"                                                       01611
           gparm2 _ #"WRITE"                                            01612
        )                                                               01613
      / "VERBOSE"                                                       01614
        gparam _ #"VERBOSE"                                             01615
      / "GROUP" <"by">                                                  01616
        gparam _ #"GROUP"                                               01617
        ( "REVERSE"                                                     01618
           gparm2 _ #"REVERSE"                                          01619
           grpopt                                                       01620
        / grpopt                                                        01621
        )                                                               01622
      / "SORT"!L2! <"by">                                               01623
        gparam _ #"SORT"                                                01624
        ( "REVERSE"                                                     01625
           gparm2 _ #"REVERSE"                                          01626
           srtopt                                                       01627
        / srtopt                                                        01628
        )                                                               01629
      ) CONFIRM                                                         01630
        seqbadd( dent, gparam)                                          01631
        seqbadd( dent, gparm2)                                          01632
        seqbadd( dent, gparm3)                                          01633
        seqbadd( dent, gparm4)                                          01634
        ;                                                               01636
  % options for grouping %
```

```
    grpopt =                                                              01637
       ( "NO" <"Grouping"> gparm3 _ #"NO"                                 01638
       / "ACCOUNT"!L2! gparm3 _ #"ACCOUNT"                                01639
       / "ARCHIVE" gparm3 _ #"ARCHIVE"                                    01640
          ( "DATE" gparm4 _ #"DATE"                                       01641
          / "STATUS" gparm4 _ #"STATUS"                                   01642
          / "TAPE" gparm4 _ #"TAPE"                                       01643
          )                                                               01644
       / "CREATION" <"Date"> gparm3 _ #"CREATION"                         01645
       / "DELETE" <"Status"> gparm3 _ #"DELETE"                           01646
       / "DUMP"!L2! gparm2 _ #"DUMP" gparm3 _ #"DUMP"                     01647
          ( "DATE" gparm4 _ #"DATE"                                       01648
          / "TAPE" gparm4 _ #"TAPE"                                       01649
          )                                                               01650
       / "FIRST" <"Version Creation"> gparm3 _ #"FIRST"                   01653
       / "LAST" <"Writer"> gparm3 _ #"LAST"                               01652
       / "NUMBER" <"of Versions to Keep"> gparm3 _ #"NUMBER"              01651
       / "PROTECT" gparm3 _ #"PROTECT"                                    01654
       / "READ" <"Date"> gparm3 _ #"READ"                                 01655
       / "WRITE" <"Date"> gparm3 _ #"WRITE"                               01656
       );                                                                 01657
    %options for sorting within groups %
    srtopt =                                                              01658
       ( "ACCOUNT"!L2! gparm3 _ #"ACCOUNT"                                01659
       / "ALPHABETICAL" gparm3 _ #"ALPHABETICAL"                          01660
       / "ARCHIVE"!L2! gparm3 _ #"ARCHIVE"                                01661
          ( "TAPE"!L2! gparm4 _ #"TAPE"                                   01663
          / "TIME" <"and Date"> gparm4 _ #"TIME"                          01664
          )                                                               01665
       / "BYTESIZE" gparm3 _ #"BYTESIZE"                                  01666
       / "CREATION" <"time and Date"> gparm3 _ #"CREATION"                01667
       / "DELETE" <"Status"> gparm3 _ #"DELETE"                           01668
       / "DUMP"!L2! gparm2 _ #"DUMP" gparm3 _ #"DUMP"                     01669
          ( "TAPE"!L2! gparm4 _ #"TAPE"                                   01670
          / "TIME" <"and Date"> gparm4 _ #"TIME"                          01671
          )                                                               01672
       / "LAST"!L2! <"Writer"> gparm3 _ #"LAST"                           01674
       / "LENGTH" <"in Bytes"> gparm3 _ #"LENGTH"                         01675
       / "NUMBER" <"of"> gparm3 _ #"NUMBER"                               01676
          ( "ACCESSES" gparm4 _ #"ACCESSES"                               01677
          / "READ" gparm4 _ #"READ"                                       01678
          / "WRITE" gparm4 _ #"WRITE"                                     01679
          / "VERSIONS" <"to keep"> gparm4 _ #"VERSIONS"                   01686
          )                                                               01680
       / "FIRST" <"Version Creation"> gparm3 _ #"FIRST"                   01681
       / "READ" <"Time and Date"> gparm3 _ #"READ"                        01682
       / "SIZE" <"in Pages"> gparm3 _ #"SIZE"                             01683
       / "WRITE" <"Time and Date"> gparm3 _ #"WRITE"                      01684
       );                                                                 01685
 END.                                                                      0726
% NLS SUPERVISOR COMMANDS %                                               0727
   SUBSYSTEM subsupervisor KEYWORD "SUPERVISOR"                           0728
     COMMAND % display subsystem stack %                                  0729
        zdspss =                                                          0730
        "<" xsublist();                                                   0731
     COMMAND % display current subsystem stack name %                     0732
```

```
      zdspcs =                                                          0733
      ">" xsubcurrent();                                                0734
   COMMAND % show syntax of a command %                                 04150
      zksyntax =                                                        05147
         "SYNTAX"!L2! <"of Command">                                    04151
            param _ NULL                                                04152
            xkbuild( param )                                            04153
            syncur _ NULL                                               04154
            synsubs                                                     04155
         CONFIRM                                                        04156
            xksyntax( syncur );                                         04157
   COMMAND % quit %                                                     0735
      zquit =                                                           0736
         "QUIT"                                                         0737
            grammar _ NULL                                              0738
            ( readconfirm()  param _ NULL xquit(param, grammar)         0739
            /  ( "TO" param _ nlssubs                                   0740
               / param _ "NLS"                                          02102
               ) CONFIRM xquit(param, grammar)                          02103
            );                                                          02183
            % note-- the rule nlssubs sets the variable grammar to
            contain a pointer to a subsystem dispatch record %          0741
   COMMAND % tjump %                                                    03810
      ztjump =                                                          03811
         "JUMP"!NOTD ! <"to">                                           03812
            (                                                           03817
               "ADDRESS"                                                03631
                  ent _ #"ITEMNOVS"                                     03632
                  dest _ DSEL( #"CHARACTER" )                           03633
                  vs _ NULL                                             03635
               / jmpcoms                                                03813
            )                                                           03814
            CONFIRM                                                     03815
            xjump(ent, dest, vs);                                       03816
   COMMAND % djump %                                                    0423
      zdjump =                                                          0424
         "JUMP"!NOTT ! <"to">                                           0425
            (                                                           0426
               lookbug() % look for a bug select %                      0427
                  ent _ #"ITEM"                                         0428
                  dest _ DSEL( #"STATEMENT" )                           0429
                  vs _ VIEWSPECS                                        01425
               / "ADDRESS" <"relative to">                              01426
                  ent _ #"ITEM"                                         01533
                  dest _ DSEL( #"STATEMENT" )                           01428
                  dest _ xjdae( dest )                                  0430
                  updcsp()                                              05190
                  vs _ VIEWSPECS                                        01429
               / jmpcoms                                                03809
            )                                                           0495
            CONFIRM                                                     0496
            xjump(ent, dest, vs);                                       0497
      jmpring = dest _ xringjump( ent );                                0498
      jmpcoms =                                                         0431
         ent _                                                          0432
            ( "ITEM"                                                    0434
```

```
        / "SUCCESSOR"                                                    0435
        / "PREDECESSOR"                                                 01532
        / "UP"                                                           0436
        / "DOWN"                                                         0437
        / "HEAD"                                                         0438
        / "TAIL"                                                         0439
        / "END" <"of Branch">                                            0440
        / "BACK"                                                         0441
        / "ORIGIN"                                                       0442
        / "NEXT"!L2!                                                     0443
        )                                                                0444
        dest _ DSEL(#"CHARACTER")                                       05205
        updcsp()                                                        05203
        vs _ VIEWSPECS                                                  05204
    /                                                                    0446
        ( ent _ "LINK"                                                   0447
          dest _ LSEL(#"LINK") vs _ NULL                                03818
        / ent _ "RETURN"                                                 0450
          CONFIRM                                                        0451
          vs _ NULL                                                     01765
          %init for moving back through ring%                            0452
             jrinit(#"RETURN")                                           0453
          PERFORM jmpring UNTIL (lookansw())                             0454
        / ent _ "FILE"                                                   0460
          (% sp()                                                        0461
             dest _ LSEL(#"OLDFILELINK")                                 03819
             vs _ VIEWSPECS   /%                                         03820
           lookbug()                                                     0463
             dest _ DSEL(#"OLDFILELINK")                                 03821
             vs _ VIEWSPECS                                              03822
          / "NAMED"                                                      03636
             ent _ #"FILENAME"                                           03823
             dest _ LSEL(#"FILE")                                        03637
             vs _ VIEWSPECS                                              03824
          / "RETURN"                                                     0469
             ent _ #"FILERETURN"                                         03825
             CONFIRM                                                     0470
             vs _ NULL                                                  01763
             %init for moving back through ring%                         0471
                jrinit(#"FILERETURN")                                    0472
             PERFORM jmpring UNTIL (lookansw())                          0473
          )                                                              0474
        / ent _ "NAME"                                                   0475
          ( lookbug()                                                    0476
          / "ANY"                                                       03826
          / "FIRST"                                                      0477
             ent _ #"FIRSTNAME"                                          03829
          / "NEXT"                                                       0478
             ent _ #"NEXTNAME"                                           03828
          / "EXTERNAL"                                                  02310
             ent _ #"EXTNAME"                                            03827
          )                                                              0479
             dest _ LSEL(#"NAME") vs _ VIEWSPECS                         03830
        / "CONTENT"                                                      0480
          ( "FIRST"                                                      0481
             ent _ #"FIRSTCONTENT"                                       03831
```

```
            / "NEXT"                                                    0482
               ent _ #"NEXTCONTENT"                                    03832
            )                                                          03833
               % display previous content %                            03834
                  dest _ xjmpcntdisp()                                  0483
               ( readrepeat()  % accept old option %                    0484
               / dest _ LSEL(#"TEXT") )                                 0485
               clearname() vs _ VIEWSPECS                              03835
         / "WORD"                                                       0487
            ( "FIRST"                                                   0488
               ent _ #"FIRSTWORD"                                      03838
            / "NEXT"                                                    0489
               ent _ #"NEXTWORD"                                       03837
            )                                                          03836
               % display previous content %                            03839
                  dest _ xjmpcntdisp()                                  0490
               ( readrepeat()  % accept old option %                    0491
               / dest _ LSEL(#"WORD") )                                 0492
               clearname() vs _ VIEWSPECS                              03840
         );                                                             0494
COMMAND % TNLS help %                                                  03926
   thelp = "HELP" ! NOTD!                                              03927
      sethflg()                                                        05142
      (myrdconfirm() param _ NULL / param _ LSEL(#"LINK"))             05143
      CLEAR <"Searching HELP file">                                    05149
      hlpinit(#"HLPCOM", param)                                        05144
      CLEAR helpt;                                                     05153
   qthelp =                                                            03930
      CLEAR <"Searching HELP file">                                    03979
      hlpinit(#"CNTLQ", NULL)                                          05145
      CLEAR helpt;                                                     05150
   helpt =                                                             03931
      setcutback() hshw cutback();                                     03932
         % This is a loop!  User must Command Delete out of this
         command %                                                     05146
   hshw =                                                              03940
      CLEAR <"Help">                                                   03941
         ( lookback()                                                  03942
            param _ qbkint()                                           03943
            param2 _ #"BACK"                                           03944
            PERFORM bckrng UNTIL (lookansw())                          03945
            CONFIRM                                                    03946
         / lkup()                                                      05089
            param _ NULL                                               05091
            param2 _ #"UP"                                             05090
         / param _ LSEL(#"LINK")                                       03947
            param2 _ #"NAME" )                                         03948
         helpshow(param2 %mode%, param %selection% );                  03949
            % sets global moremenu FALSE if finished; TRUE if not
            and more is to be displayed %                              03950
COMMAND % DNLS help %                                                  03951
   dhelp = "HELP" ! NOTT!                                              03952
      sethflag()                                                       05141
      (myrdconfirm() param _ NULL / param _ LSEL(#"LINK"))             05135
      CLEAR <"Searching HELP file">                                    05134
      hlpinit(#"HLPCOM", param)                                        05139
```

```
      CLEAR helpd;                                                    05152
   qdhelp =                                                           03954
      CLEAR <"Searching HELP file">                                   03978
      hlpinit(#"CNTLQ", NULL)                                         05140
      CLEAR helpd;                                                    05151
   helpd =                                                            03955
      setcutback() drpthelp cutback();                                03956
         % This is a loop!  User must Command Delete out of this
         command %                                                    05138
   drpthelp =                                                         03957
      hchk dshw;                                                      03958
   hchk =                                                             03935
      checkmore()  % Check moremenu %                                 03936
         CLEAR <"do you want to see the rest of the menu?">           05099
         answ()                                                       03938
         helpshow(#"NEXT", NULL) (hchk / true())                      05098
      /  rstmor();                                                    03939
         % reset moremenu (did not want next menu); close
         sequence.  moremenu always FALSE, sequence qsw closed
         upon exit from this rule. %                                  05148
   dshw =                                                             03959
      CLEAR <"Help">                                                  03960
         % ( mylookbug()                                             03961
            param _ DSEL(#"STATEMENT")                                03962
            param2 _ #"MENU"                                          03963
         / lookrpt()                                                  03964
            param _ DSEL(#"WORD")                                     03965
            param2 _ #"NAME" %                                        03966
         ( lookback()                                                 03967
            param _ gbkint()                                          03968
            param2 _ #"BACK"                                          03969
            PERFORM bckrng UNTIL (lookansw())                         03970
            CONFIRM                                                   03971
         / lkup()                                                     05092
            param _ NULL                                              05093
            param2 _ #"UP"                                            05094
         / looknxt()                                                  05095
            param _ NULL                                              05096
            param2 _ #"NEXT"                                          05097
         / dumprompt() param _ LSEL(#"LINK")                          03972
            param2 _ #"NAME"                                          03973
         )                                                            03974
         helpshow(param2 %mode% , param %selection% );                03975
            % sets global moremenu FALSE if finished; TRUE if not
            and more message is to be displayed %                     03976
      bckrng = param _ xhlpring(param);                               03977
COMMAND % goto %                                                       0743
   zgoto =                                                             0744
      "GOTO" <"subsystem">                                             0745
         % note-- the rule nlssubs sets the variable grammar to
         contain a pointer to a subsystem dispatch record %            0746
         grammar _ NULL                                                0747
         param _ (nlssubs / "TENEX" )                                  0748
         CONFIRM xgoto(param, grammar, FALSE);                         0749
COMMAND % execute %                                                    0750
   zexecute =                                                          0751
```

```
        "EXECUTE" <"command in">                                        0752
           % note-- the rule nlssubs sets the variable grammar to
           contain a pointer to a subsystem dispatch record %           0753
           grammar _ NULL                                               0754
           param _ nlssubs                                              0755
            xgoto(param, grammar, TRUE);                                0756
    COMMAND % comment %                                                 0698
       zcomment =                                                       0699
          ";"                                                           0700
             xcomment();                                                0701
  END.                                                                  0757
% USER PROGRAMMING SUBSYSTEM COMMANDS %                                0758
   SUBSYSTEM subprograms KEYWORD "PROGRAMS"                             0759
     uprogtypes =                                                       0760
       ("CONTENT" <"analyzer program">/ "SORT" <"key extractor
       program">/ "SEQGEN"!L2! <"program"> );                           0761
     COMMAND %show%                                                     0771
       zpshow =                                                         0772
          "SHOW"                                                        0773
             ( "STATUS" <"of programs buffer">                          01817
                CONFIRM xpshow( )                                       0774
             / "TENEX" <"Subsystem Status">                             01818
                CONFIRM xpshtn( )                                       01819
             );                                                         01820
     COMMAND %set%                                                      0762
       zpset =                                                          0763
          "SET"!L2!                                                     0764
             ent _                                                      0765
                ( "BUFFER" <"size to">                                  0766
                   source _ LSEL(#"NUMBER")                             0767
                / "NDDT" <"control-h">                                  0768
                   source _ NULL )                                      0769
             CONFIRM xpset(ent, source );                               0770
     COMMAND %run%                                                      01790
       zprun =                                                          03332
          "RUN"                                                         03333
             ( "PROGRAM"                                                03334
                source _ LSEL(#"NAME")                                  03335
                CONFIRM xprun( source )                                 03336
             / "TENEX" <"Subsystem">                                    03337
                source _ LSEL(#"OLDFILELINK") CONFIRM                   03338
                CLEAR <"Output to File or Teletype?">                   03339
                   ( "FILE" param _ LSEL(#"NEWFILELINK")                03340
                   / "TELETYPE" param _ FALSE                           03341
                   ) CONFIRM                                            03342
                CLEAR <"Input Mode?">                                   03343
                   param2 _                                             03344
                      ( "FROM"! 3! <"FILE:">                            03345
                         param3 _ LSEL(#"OLDFILELINK")                  03346
                      / "INTERACTIVE"! 1! <"Termination Character:">
                                                                        03347
                         param3 _ LSEL(#"CHARACTER")                    03348
                      / "TYPEAHEAD"! 2!                                 03349
                         param3 _ LSEL(#"TEXT")                         03350
                      / "NO"! 4! <"Input">                              03351
                         param3 _ FALSE                                 03352
```

```
                    ) CONFIRM                                          03353
                CLEAR <"Wait for Completion?">                         03354
                   ( "YES" param4 _ TRUE                               03355
                   / "NO" param4 _ FALSE                               03356
                   ) CONFIRM                                           03357
                CLEAR <"Go?">                                          03358
                CONFIRM xprunt( source, param, param2, param3, param4
                )                                                      03359
             );                                                        03360
COMMAND %reset%                                                         0775
   zpreset =                                                            0776
      "RESET"!L2!                                                       0777
         param _ NULL  param2 _ NULL                                   01512
         ent _                                                          0778
            ( "BUFFER" <"size">                                         0779
            / "NDDT" <"control-h">                                      0780
            % commented out (never worked anyway)                      05208
               / "PARSERULE"                                           01509
                  <"named">   param _ LSEL(#"NAME")                    01510
                  <"in rel-file named"> param2 _
                  LSEL(#"OLDFILELINK")                                  01511
            %                                                          05209
            )                                                          05210
         CONFIRM xpreset( ent, param, param2 );                         0781
%   Commented out (never worked anyway); percents doubled              05207
   COMMAND %%replace%%                                                 01498
      zpreplace =                                                      01499
         "REPLACE"!L2!                                                 01500
            "PARSERULE"                                                01502
               <"named">   param _ LSEL(#"NAME")                       01505
               <"in rel-file named"> param2 _ LSEL(#"OLDFILELINK")
                                                                       01506
               CLEAR <"by new parserule named">                        01507
               param3 _ LSEL(#"NAME")                                  01508
            CONFIRM xpreplace( param, param2, param3 );                01504
%                                                                      05206
COMMAND %load%                                                         03417
   zpload =                                                            03418
      "LOAD" "PROGRAM"                                                 03419
         xpsetf()                                                      03420
         source _ LSEL(#"OLDFILELINK")                                 03421
         CONFIRM xpload( source );                                     03422
COMMAND %kill%                                                         01813
   zpkill =                                                            01814
      "KILL" "TENEX" <"Subsystem">                                     01815
      CONFIRM xpkill();                                                01816
COMMAND %institute%                                                     0792
   zpinstitute =                                                        0793
      "INSTITUTE" "PROGRAM"                                             0794
         source _ LSEL(#"NAME") <"as">                                  0795
         ent _  uprogtypes                                              0796
         CONFIRM xpinstitute( source, ent );                            0797
COMMAND %detach%                                                       03508
   zpdetach =                                                          03509
      "DETACH"!L2!                                                     03510
         "SUBSYSTEM"                                                   03511
```

```
                grammar _ NULL                                          03512
                param _ nlssubs                                         03513
                CONFIRM xpdetach( param, grammar );                     03514
   COMMAND %deinstitute%                                                 0798
      zpdeinstitute =                                                    0799
         "DEINSTITUTE"!L2!                                               0800
            ent _  uprogtypes                                            0801
            CONFIRM xpdeinstitute( ent );                                0802
   COMMAND %delete%                                                      0803
      zpdelete =                                                         0804
         "DELETE"                                                        0805
            ent _                                                        0806
               ( "ALL" <"programs in buffer">                            0807
               / "LAST" <"program in buffer"> )                          0808
               CONFIRM xpdelete( ent );                                  0809
   COMMAND %compile%                                                     0810
      zpcompile =                                                        0811
         "COMPILE"                                                       0812
            sent _                                                       0813
               ( ("FILE" %/ "ASSEMBLER" <"file">% )                      0814
                  <"at"> source _ DSEL(#"STATEMENT")                     0815
                  <"using"> param _ LSEL(#"OLDFILELINK")                 0816
                  <"to file"> namfil _ LSEL(#"NEWFILELINK")              0817
               / "PROCEDURE" <"at">                                      0818
                  param _ NULL  namfil _ NULL                            0819
                  source _ DSEL(#"STATEMENT")                            0820
               / "L10" <"user program at">                             03151
                  param _ NULL  namfil _ NULL                            03152
                  source _ DSEL(#"STATEMENT")                            03153
               / "CONTENT" <"pattern">                                   01759
                  param _ NULL  namfil _ NULL                            01760
                  source _ LSEL(#"CHARACTER") )                          01761
               CONFIRM xpcompile( sent, source, param, namfil );
                                                                        0821
   COMMAND %attach%                                                      03515
      zpattach =                                                         03516
         "ATTACH"                                                        03517
            "SUBSYSTEM"                                                  03518
               grammar _ NULL                                            03519
               param _ allsubs                                           03520
               CONFIRM xpattach( param, grammar );                       03521
END.                                                                     0828
% SENDMAIL SUBSYSTEM COMMANDS %                                         0829
  SUBSYSTEM subsendmail KEYWORD "SENDMAIL"                               0830
    COMMAND %structure entities%                                        02676
      zjstructure =                                                     02677
         param _ ("STATEMENT"!L2!/"GROUP"!L2!/"PLEX"!L2!/"BRANCH")
                                                                        0961
            <"at"> param2 _ SSEL(param) CONFIRM jsetfield(param,
            param2) ;                                                    0962
    COMMAND %author%                                                    01862
      zjauthor =                                                        01892
         "AUTHORS"                                                       0846
            param _ LSEL(#"IDENTLIST")                                   0847
            CONFIRM jsetfield(#"AUTHORS", param) ;                       0848
    COMMAND %comment%                                                   02650
```

```
   zjcomment =                                                    02651
      "COMMENT"                                                    0851
      param _ LSEL(#"TEXT")                                        0852
      CONFIRM jsetfield(#"COMMENT", param) ;                       0853
COMMAND   %copy content%                                          05179
   zjcopycontent =                                                05180
      "COPY"!L2! <"content of mail to follow ">                   05181
         dest _ DSEL(#"STATEMENT")                                05182
         level _ LEVADJ                                           05183
         xjcopycontent(dest, level);                              05184
COMMAND %distribute%                                              02652
   zjdistribute =                                                 02653
      "DISTRIBUTE" <"for">                                        03405
         param _ ("ACTION"/"INFORMATION" <"Only">)                03406
         <"to"> param2 _ LSEL(#"IDENTLIST") CONFIRM               03409
         jsetfield(param, param2) ;                               03410
COMMAND %expedite%                                                02654
   zjexpedite =                                                   02655
      "EXPEDITE"!L2!                                              02181
         CONFIRM jsetfield(#"EXPEDITE", answ()) ;                 02182
COMMAND %file%                                                    02656
   zjfile =                                                       02657
      param _ "FILE" param2 _ DSEL(#"CHARACTER")                   0859
      CONFIRM jsetfield(param, param2) ;                           0863
COMMAND %forward mail%                                            02635
   zjforward =                                                    02636
      "FORWARD"!L2! <"item number">                               02637
         param _ LSEL(#"NUMBER")                                  02638
         <"for"> param2 _ ("ACTION" / "INFORMATION" <"only">)
                                                                  02729
         <"to"> param3 _ LSEL(#"IDENTLIST")                       02730
         CONFIRM xjforward(param, param2, param3);                02639
COMMAND %initialize specification%                                02968
   zjispecs =                                                     02969
      "INITIALIZE"!L2!                                            02970
         <"specifications">                                       02971
         CONFIRM xjzapworfil() ;                                  02972
COMMAND %insert%                                                  02660
   zjinsert =                                                     05154
      "INSERT"!L2!                                                05155
         ( "LINK" <"to follow">                                   05156
            param _ DSEL(#"VISIBLE")                              05157
            CONFIRM jsetfield(#"INSERT", param)                   05158
         / "STATUS" <"form to follow">                            05159
            param _ DSEL(#"STATEMENT")                            05160
            param2 _ LEVADJ                                       05161
            CONFIRM xjinsstatus(param, param2)                    05162
         / "RECORD" <"for ident">                                 05163
            param _ LSEL(#"IDENT")                                05164
            <"as a statement to follow">                          05165
            param2 _ DSEL(#"STATEMENT")                           05166
            param3 _ LEVADJ                                       05167
            CONFIRM xjinsrecord(param, param2, param3)            05168
         );                                                       05169
COMMAND %interrogate%                                             02662
   zjinterrogate =                                                02663
```

```
        "INTERROGATE"                                                  0874
           CONFIRM                                                    01895
           CLEAR <"distribute for action to:">                         0890
              param _ LSEL(#"IDENTLIST")                              03154
              jsetfield(#"ACTION", param)                             01917
           CLEAR <"distribute for information-only to:">              03035
              param _ LSEL(#"IDENTLIST")                              03155
              jsetfield(#"INFORMATION", param)                        03036
           CLEAR <"title:"> param _ LSEL(#"TEXT")                      0889
              jsetfield(#"TITLE", param)                              01916
           CLEAR <"type of source:">                                  05191
           param _          %pull outside alts so backup over source
           type will work -- CLI glitch causing the problem%          0875
              ( "MESSAGE"  % NOTE:  same value as #"TEXT" %           0883
                 param2 _ LSEL(#"TEXT")                               0884
              / "FILE"                                                0878
                 param2 _ DSEL(#"CHARACTER")                          0880
              % structures type broken out so backup will work --
              CLI glitch %                                            05199
              / "STATEMENT"                                           0876
                 <"at"> param2 _ SSEL(#"STATEMENT")                   0877
              / "BRANCH"                                              05193
                 <"at"> param2 _ SSEL(#"BRANCH")                      05194
              / "GROUP"                                               05195
                 <"at"> param2 _ SSEL(#"GROUP")                       05196
              / "PLEX"                                                05197
                 <"at"> param2 _ SSEL(#"PLEX")                        05198
              %                                                       03522
              / "OFFLINE" <"item">                                    0885
                 <"located at"> param2 _ LSEL(#"TEXT")                0886
              %                                                       03523
              )                                                       0887
              jsetfield(param, param2)                                0888
           CLEAR <"show status?"> (answ() xjstatus() / DUMMY)         0891
           CLEAR <"send the mail now?">                               05185
              (answ()                                                 05186
                 <"really?"> needconfirm _ TRUE                       05187
                 CONFIRM xjdoit()                                     05188
              / DUMMY)                                                05189
           xdummy();  % to get around a CLI bug %                     05125
     COMMAND %keywords%                                               02664
        zjkeywords =                                                  02665
           "KEYWORDS"                                                  0901
              param _ LSEL(#"TEXT")                                    0902
              CONFIRM jsetfield(#"KEYWORDS", param) ;                  0903
     COMMAND %message%                                                02672
        zjmessage =                                                   02673
           "MESSAGE"                                                   0906
              param _ LSEL(#"TEXT") CONFIRM                            0907
              jsetfield(#"TEXT", param) ;                             01918
     COMMAND %number%                                                 02670
        zjnumber =                                                    02671
           "NUMBER"                                                    0910
              ("ASSIGN" CONFIRM                                        0911
                 param _ xjreserve(#"JOURNAL", NULL, FALSE) /         02995
              "PREVIOUSLY" <"Reserved"> param _ LSEL(#"NUMBER") )      0912
```

```
            CONFIRM jsetfield(#"NUMBER", param) ;                    0913
   COMMAND %obsolets%                                                02668
      zjobsolets =                                                   02669
         "OBSOLETES"                                                  0916
            <"item number(s)">  param _ LSEL(#"TEXT")               01896
            CONFIRM jsetfield(#"OBSOLETES", param) ;                  0918
   COMMAND %offline item%                                            02658
      zjoffline =                                                    02659
         "OFFLINE"!L2!                                                0870
            <"item -- located at"> param _ LSEL(#"TEXT") CONFIRM
            jsetfield(#"HARDCOPY", param) ;                           0871
   COMMAND %process%                                                 02666
      zjprocess =                                                    02667
         "PROCESS"                                                    0926
            <"sendmail form at">                                    01897
            param _ DSEL(#"STATEMENT")                                0927
            CONFIRM                                                   0928
            xjprocess(param) ;                                        0929
   COMMAND %private%                                                 03056
      zjprivate =                                                    03057
         "PRIVATE"!L2!                                              03058
            CONFIRM jsetfield(#"PRIVATE", TRUE) ;                   03059
   COMMAND %public%                                                  03060
      zjpublic =                                                     03061
         "PUBLIC"!L2!                                               03062
            CONFIRM jsetfield(#"PUBLIC", TRUE) ;                    03063
   COMMAND %reserve number%                                          01884
      zjreserve =                                                    01903
         "RESERVE" (                                                  0932
            param _                                                   0933
               ("JOURNAL" / "XDOC")                                   0934
               <"numbers -- how many?"> param2 _ LSEL(#"NUMBER")
                                                                      0935
               CLEAR <"insert the number list?">                      02991
                  (answ() <"to follow"> param3 _ DSEL(#"VISIBLE") /
                                                                      02985
                  <"No."> param3 _ NULL)                              02986
               CONFIRM xjreserve(param, param2, param3) /             02990
            param _ "RFC" <"number"> CONFIRM                          0936
               CLEAR <"title"> param _ LSEL(#"TEXT")                  0937
               CLEAR <"author"> param2 _ LSEL(#"TEXT")                0938
               CLEAR <"distribute to"> param3 _ LSEL(#"IDENTLIST")
                                                                      0939
               CLEAR <"online document?"> source _ answer()           0940
               CLEAR <"show status?">                                 0941
                  (answ() xjrfcshow(param, param2, param3, source) /
                  DUMMY)                                              02994
               CLEAR <"insert the number list?">                      02987
                  (answ() CLEAR <"to follow"> param4 _
                  DSEL(#"VISIBLE") /                                  02988
                  <"No."> param4 _ NULL  DUMMY)                       02989
               CONFIRM                                                0942
               xjrfcreserve(param, param2, param3, source, param4)
                                                                      0943
         ) ;                                                          0944
   COMMAND %rfc%                                                     02674
```

```
      zjrfc =                                                          02675
         "RFC"!L2!                                                      0947
            <"number">  param _ LSEL(#"NUMBER")                        01898
            CONFIRM jsetfield(#"RFC", param) ;                          0949
   COMMAND %send%                                                       02680
      zjsend =                                                         02681
         "SEND" <"the mail"> CONFIRM                                    0856
         <"really?"> needconfirm _ TRUE CONFIRM                        05200
         xjdoit() ;                                                    05201
   COMMAND %show status%                                                02678
      zjstatus =                                                       02679
         "SHOW"!L2!                                                     0952
            ( "STATUS" CONFIRM xjstatus ()                             03783
            / "RECORD"  <"for ident">                                  03785
               param _ LSEL(#"IDENT")                                  05211
               param2 _ #"UNDELETE"                                    05174
               CONFIRM xjistatus(param, param2)                        05175
            );                                                         05173
   COMMAND %subcollections%                                             02682
      zjsubcollections =                                               02683
         "SUBCOLLECTIONS"!L2!                                           0956
            param _ LSEL(#"IDENTLIST")                                  0957
            CONFIRM jsetfield(#"SUBCOLLECTIONS", param) ;                0958
   COMMAND %title%                                                      02686
      zjtitle =                                                        02687
         "TITLE"                                                        0965
            param _ LSEL(#"TEXT")                                       0966
            CONFIRM jsetfield(#"TITLE", param) ;                        0967
   COMMAND %update%                                                     02684
      zjupdate =                                                       02685
         "UPDATE"!L2!                                                   0970
            <"to item number(s)"> param _ LSEL(#"TEXT")                01902
            CONFIRM jsetfield(#"UPDATE", param) ;                       0971
   COMMAND %unrecorded%                                                 02908
      zjunrecorded =                                                   02909
         "UNRECORDED"                                                   02910
            CONFIRM jsetfield(#"UNRECORDED", answ()) ;                  02912
   INITIALIZATION %get work file%                                       0831
      zjinit =                                                          0832
         xwarning() xjloaworfil();                                     02967
   END.                                                                 0972
% USER-OPTIONS SUBSYSTEM COMMANDS %                                    01145
   SUBSYSTEM subuseroptions KEYWORD "USEROPTIONS"                       01146
      INITIALIZATION                                                    01821
         zuoinit=                                                       01822
            xuoinit();                                                  01823
      TERMINATION                                                       01824
         zuoterm=                                                       01825
            xuoterm();                                                  01826
      COMMAND %control characters%                                      01147
         zuocharacters =                                                01148
            "CONTROL"!L2! <"characters for terminal">                   01149
            param _ devopt                                              01836
            CONFIRM                                                     01840
            CLEAR <"control character">                                 01841
            param1 _                                                    01150
```

```
            ( "CA" / "CD"!L2! / "RPT" / "INSERT" / "BC" / "BW"!L2! /
            "BS"!L2! / "LITESC" / "IGNORE"!L2! / "SC" / "SW"!L2! /
            "TAB"!L2!)                                                  01151
         <"character(s)"> param2 _ LSEL(#"TEXT")                        01152
         <"echo as"> param3 _ LSEL(#"TEXT")                             01153
         CONFIRM xuocontchars(param, param1, param2, param3);            01154
            %if the user types n chars in first lit, he should type
            only one char in second lit -- subsequent chars will
            ignored%                                                    01155
   COMMAND %current content length%                                     01156
      zuocurcon =                                                       01157
         "CURRENTCONTEXT" <"length">                                    01158
            param _ LSEL(#"NUMBER") CONFIRM xuocurcon(param);            01159
            %number of chars to be printed for each side of CMi in
            slash command%                                              01160
   COMMAND %display areas%                                              05042
      zuodisplay =                                                      05043
         "DISPLAY" <"area">                                             05119
            param1 _                                                    05120
               ( "RIGHT" CLEAR <"right margin max for any area is
               column">                                                 05121
               / "WRAPAROUND" CLEAR <"wraparound margin is column">
                                                                        05122
               )                                                        05123
               param2 _ LSEL(#"NUMBER")                                 05124
         CONFIRM xuodisplay(param1, param2);                            05047
   COMMAND %entry subsystem%                                            03537
      zuoentry =                                                        03538
         param1 _ "ENTRY"!L2!                                           03539
         param _ NULL                                                   03540
         grammar _ NULL                                                 03541
            (param2 _ "SUBSYSTEM" param _ allsubs  %stores into the
            varialble grammar%                                          03542
            / param2 _ "PROGRAM" param _ LSEL(#"OLDFILELINK"))          03543
         CONFIRM xuoinclude( param, param1,param2 );                    03544
   COMMAND %exclude subsystem/program%                                  03545
      zuoexclude =                                                      03546
         "EXCLUDE"!L2!                                                  03547
            (param2 _ "SUBSYSTEM" param _ allsubs  %stores into the
            varialble grammar%                                          03548
            / param2 _ "PROGRAM" param _ LSEL(#"OLDFILELINK"))          03549
            CONFIRM xuoexclude( param,param2 );                         03550
   COMMAND %external names link file address%                           03175
      zuoextn =                                                         03176
         "EXTERNAL"!L2! <"Names Link File Address:">                    03177
            param _ LSEL(#"LINK")                                       03178
            CONFIRM xuoextn( param );                                   03179
   COMMAND %feedback%                                                   01161
      zuofeedback =                                                     01162
         "FEEDBACK"                                                     01163
            param2 _ NULL                                               01164
            param _                                                     01165
               ("VERBOSE" /                                             01166
               "TERSE" /                                                01167
               ( ("LENGTH" / "INDENTING")                               01168
                  param2 _ LSEL(#"NUMBER")))                            01169
```

```
         CONFIRM xuofeedback(param, param2);                          01170
   COMMAND %herald%                                                   01171
      zuohearald =                                                    01172
         "HERALD"!L2!                                                 01173
            param2 _ NULL                                             01174
            param _ ("TERSE" /                                        01175
               "VERBOSE" /                                            01176
               "LENGTH" param2 _ LSEL(#"NUMBER"))                     01177
            CONFIRM xuohearald(param, param2);                        01178
   COMMAND %include subsystem/program%                                03551
      zuoinclude =                                                    03552
         param1 _ "INCLUDE"                                           03553
         grammar _ NULL                                               03554
         param _ NULL                                                 03555
            (param2 _ "SUBSYSTEM" param _ allsubs  %stores into the
            varialble grammar%                                        03556
            / param2 _ "PROGRAM" param _ LSEL(#"OLDFILELINK"))        03557
         CONFIRM xuoinclude( param, param1,param2 );                  03558
   %COMMAND level adjust                                              03017
      zuolevel =                                                      03018
         "LEVELADJUST"                                                03019
            param _ switch                                            03020
            CONFIRM                                                   03022
            xuolevel(param);%                                         03021
   COMMAND %name delimiter defaults%                                  04228
      zuonamed =                                                      04229
         "NAME" <"delimiter defaults">                                04230
            <"left delimiter"> param1 _ LSEL(#"CHARACTER")            04231
            <"right delimiter"> param2 _ LSEL(#"CHARACTER")           04232
            xuonamed(param1,param2);                                  04233
   COMMAND %output%                                                   05048
      zuooutput =                                                     05049
         "OUTPUT"                                                     05050
            param _ "QUICKPRINT"                                      05051
               param1 _ "RIGHT" <"margin is column">                  05052
                  param2 _ LSEL(#"NUMBER")                            05053
            CONFIRM xuooutput(param, param1, param2);                 05054
   COMMAND %printing parameters%                                      01186
      zuoprint =                                                      01187
         "PRINTOPTIONS"!L2! (                                         01188
            param _                                                   01189
               (                                                      01190
                  ("RIGHT" / "LEFT") <"margin is column"> /           01191
                  "BOTTOM" <"margin is line"> /                       01192
                  "PAGE" <"size is (lines)"> /                        01193
                  "INDENTING" <"per level"> )                         01194
               param2 _ LSEL(#"NUMBER") /                             01195
            param _ "TAB" <"stop settings"> param2 _ LSEL(#"TEXT") )
                                                                      01196
               %TAB text is of form
               c       c       c       c       c               c       ,
               where c stands for any char except SP, or of the form
               8,16,24,... , where the number represents the column
               position (starting from 1 not 0) for the next tab
               stop%                                                  01197
            CONFIRM xuoprint(param, param2);                          01198
```

```
COMMAND %prompt mode%                                                   01199
   zuoprompt =                                                          01200
      "PROMPT"                                                          01201
      param _ ("OFF" / "PARTIAL" / "FULL") CONFIRM                      01202
      xuoprompt(param);                                                 01203
COMMAND %recognition mode%                                              01204
   zuorecognition =                                                     01205
      "RECOGNITION" <"mode"> param _                                    01206
         (rmode param2 _ NULL /                                         01207
         "TERSE"                                                        01208
            <"secondary mode"> param2 _ (rmode / "TERSE"))              01209
      CONFIRM xuorecognition(param, param2);                            01210
      rmode =                                                           01211
         ("ANTICIPATORY" / "DEMAND" / "FIXED");                         01212
COMMAND %reset%                                                         03559
   zuoreset =                                                           03560
      "RESET"!L2!                                                       03561
         param2 _ NULL                                                  03562
         param1 _ NULL                                                  05055
         param _                                                        03563
            ( "ALL"                                                     03564
            / "CONTROL"!L2! <"characters for terminal">                 03565
               param2 _                                                 03566
                  ( devopt                                              03567
                  / "ALL" )                                             03568
            / "CURRENTCONTEXT" <"length">                               03569
            / "DISPLAY"!L2!                                             05114
               param2 _                                                 05115
                  ( "RIGHT" <"margin max">                              05116
                  / "WRAPAROUND" <"margin">                             05117
                  )                                                     05118
            / "ENTRY"!L2! <"Subsystem">                                 03570
            / "EXTERNAL" <"Names Link File Address">                    03571
            / "FEEDBACK"                                                03572
               param2 _                                                 03573
                  ("MODE" / "LENGTH" / "INDENTING")                     03574
            / "HERALD"                                                  03575
               param2 _ ("MODE" / "LENGTH")                             03576
            / "DEFAULT" <"Subsystems and Programs">                     03577
            / "NAME" <"delimiters">                                     04235
            / ("RETURN" / "FILERETURN"!L2!)                             03578
               <"ring entries">                                         03580
            / "OUTPUT"                                                  05058
               param2 _ "QUICKPRINT"                                    05059
                  param1 _ "RIGHT" <"margin">                           05060
            / "PRINTOPTIONS"!L2!                                        03581
               param2 _                                                 03582
                  ( ("RIGHT" / "LEFT" / "BOTTOM")                       03583
                     <"margin">                                         03584
                  / "PAGE" <"size">                                     03585
                  / "INDENTING" <"per level">                           03586
                  / "TAB" <"stop settings">                             03587
                  )                                                     03588
            / "PROMPT" <"mode">                                         03589
            / "RECOGNITION"!L2! <"mode">                                03590
            / "SPACE"!L2! <"for tabs to OFF">                           05082
```

```
            / "STARTUP" <"Commands Branch Address">                    03592
            / "UNIVERSAL" <"Subsystem">                                03591
            / "VIEWSPECS"                                              03593
            )                                                          03594
      CONFIRM xuoreset(param, param2, param1);                         03595
COMMAND %return/filereturn ring size%                                  01179
   zuoringsize =                                                       01180
      param _ ("RETURN"!L2! / "FILERETURN"!L2!)                        01182
      <"ring entries">                                                 01183
         param2 _ LSEL(#"NUMBER")                                      01184
      CONFIRM xuoringsize(param, param2);                              01185
COMMAND %show%                                                         03596
   zuoshow =                                                           03597
      "SHOW"                                                           03598
         param1 _ NULL                                                 03599
         param _                                                       03600
            ("ALL"                                                     03601
            / ( "CONTROL"!L2! <"characters for terminal">              03602
               param1 _                                                03603
                  ("ALL"                                               03604
                  / devopt)                                            03605
               )                                                       03606
            / "CURRENTCONTEXT"                                         03607
            / "DISPLAY"!L2! <"area">                                   05061
            / "ENTRY"!L2! <"subsystem">                                03608
            / "EXTERNAL" <"Names Link File Address">                   03609
            / "DEFAULT" <"Subsytems and Programs">                     03610
            / "FEEDBACK"                                               03611
            / "HERALD"                                                 03612
            / "RETURN"!L2! <"ring sizes">                              03613
            / "LEVELADJUST"                                            03614
            / "NAME" <"delimiter defaults">                            04234
            / "OUTPUT"                                                 05062
            / "PRINTOPTIONS"!L2!                                       03615
            / "PROMPT"                                                 03616
            / "RECOGNITION"                                            03617
            / "SPACE" !L2! <"for tabs status">                         03618
            / "STARTUP" <"Commands Branch Address">                    05083
            / "UNIVERSAL" <"Subsystem">                                03619
            / "VIEWSPECS"                                              03620
            )                                                          03621
      CONFIRM xuoshow(param, param1);                                  03622
COMMAND %space for tabs/backspace/termination%                         05063
   zuospace = "SPACE" !L2! <"for tabs">                                05064
      sent _ FALSE                                                     05084
      source _ FALSE                                                   05085
      dent _ FALSE                                                     05088
      ( lookconfirm() dent _ TRUE                                      05065
      / "YES" dent _ TRUE                                              05066
      / "NO" dent _ FALSE                                              05067
      / "BACKSPACE" <"automatically">                                  05068
         ( lookconfirm()                                               05069
            dent _ TRUE                                                05070
            sent _ TRUE                                                05071
         / "TERMINATION" <"characters">                                05072
            dent _ TRUE                                                05086
```

```
                sent _ TRUE                                                 05087
                CLEAR                                                       05073
                <"Termination characters besides CR and TAB">               05075
                source _ LSEL(#"TEXT")                                      05076
             )                                                              05077
          )                                                                 05078
          CONFIRM                                                           05079
          xuospfortab(dent, sent, source);                                  05080
       COMMAND %startup%                                                    03156
          zuostartup =                                                      03157
             "STARTUP"!L2! <"Commands Branch Address">                      03158
                param _ LSEL(#"LINK")                                       03159
                CONFIRM xuostup( param );                                   03160
       %COMMAND universal subsystem                                         03623
          zuouniversal =                                                    03624
             param1 _ "UNIVERSAL"!L2!                                       03625
             grammar _ NULL                                                 03626
             param _ NULL                                                   03627
                (param2 _ "SUBSYSTEM" param _ allsubs  %%stores into the
             varialble grammar%%                                            03628
                / param2 _ "PROGRAM" param _ LSEL(#"OLDFILELINK"))          03629
             CONFIRM xuoinclude( param, param1,param2 );%                   03630
       COMMAND %viewspecs%                                                  01213
          zuoviewspecs =                                                    01214
             "VIEWSPECS"                                                    01215
                param1 _ NULL                                               03016
                param _                                                     03015
                   ( "DEFAULT"                                              03013
                      param1 _ VIEWSPECS                                    01216
                   %/  switch %) % removed until it functions correctly%
                                                                            03014
                CONFIRM xuoviewspecs(param, param1);                        01217
       END.                                                                 01248
% UTILITY SUBSYSTEM COMMANDS %                                             03068
    SUBSYSTEM subxxx KEYWORD "XXX"                                          03069
       %INITIALIZATION                                                      03070
          zxinit=                                                           03071
             xutilinit();                                                   03072
       TERMINATION                                                          03073
          zxterm=                                                           03074
             xuquit();%                                                     03075
       COMMAND %run utility functions %                                     03080
          zxrun =                                                           03081
             "RUN"                                                          03082
                ( "JOURNAL"                                                 03083
                   CONFIRM                                                  03084
                   xjoutil()                                                03085
                / ("TASKS"                                                  03086
                   param _ #"FILE"                                          03087
                   ( readconfirm()                                          03088
                   /  param _                                               03089
                      ( "DETACH"!NOTD !                                     03090
                      / "FILE"                                              03091
                      / "TTY")                                              03092
                      CONFIRM)                                              03093
                   xcutil (param)));                                        03094
```

```
   COMMAND %set utility options %                                          03095
      zxset =                                                              03096
         "SET"                                                             03097
            ( "JOURNAL" <"options"> setbkup()                              03098
               param _                                                     03099
               ( "AUTO" <"hard copy distribution">                         03100
               / "CONTINUE"                                                03101
               / "ON" <"Line Journal Delivery">                            03102
               / "RECOVER" <"files">                                       03103
               / "SLINKER"                                                 03104
               / "UPDATE" <"initial files">)                               03105
                  CONFIRM                                                  03106
                  xsetjou(param)                                           03107
            / "PARTIAL" <"Delivery options">                               03108
               param _                                                     03109
               ( "CLEAR" <"options">                                       03110
                  param1 _ NULL                                            03111
               / "IDENTS" <"for on-line delivery">                         03112
                  param1 _ LSEL(#"IDENTLIST")                              03113
               / "FILES" <"for input">                                     03114
                  param1 _ LSEL(#"TEXT")                                   03115
               / "NUMBER" <"of documents separated by spaces">             03116
                  param1 _ LSEL(#"TEXT")   )                               03117
                  CONFIRM                                                  03118
                  xsetpar(param1, param)                                   03119
            / "PRIORITY"!L2!                                               03120
               <"Value">                                                   03121
               param _ LSEL(#"NUMBER")                                     03122
               CONFIRM                                                     03123
               xpriority(param)                                            03124
            / "LOAD" <"average">                                           03125
               param _ LSEL(#"NUMBER")                                     03126
               CONFIRM                                                     03127
               xsetld(param));                                             03128
   COMMAND %unlock%                                                        03129
      zxjunlock =                                                          03130
         "UNLOCK" "JOURNAL" <"password:">                                  03131
         param _ LSEL(#"VISIBLE")                                          03132
         CONFIRM xjlock(param, FALSE);                                     03133
   % NOT IMPLEMENTED   (dash substituted for percent)                      03524
   COMMAND -print-                                                         03134
      zxjprint =                                                           03135
         "PRINT" "HARDCOPY" -wheel()-                                      03136
         <"password:"> param _ LSEL(#"VISIBLE") CONFIRM                    03137
         xjpriharcop(param);                                               03138
   %                                                                       03525
   COMMAND %lock%                                                          03139
      zxjlock =                                                            03140
         "LOCK"!L2! "JOURNAL" <"password:">                                03141
         param _ LSEL(#"VISIBLE")                                          03142
         CONFIRM xjlock(param, TRUE);                                      03143
   COMMAND %load%                                                          03920
      zxload =                                                             05100
         "LOAD"                                                            05101
            (ent _ "BUSY" <"file">                                         05102
               namfil _ LSEL(#"OLDFILELINK") CONFIRM                       05103
```

```
                xload(ent, namfil)                                    05104
              / "JNLDEL" <"into program buffer">                      05105
                CONFIRM xjnlload()                                    05106
              );                                                      05107
    COMMAND % enter ddt %                                             03250
      zxddt =                                                         03251
        "TENEX" <"Ddt">                                               03252
        CONFIRM xxddt();                                              03253
    COMMAND %detach%                                                  03361
      zxdetach =                                                      03362
        "DETACH"                                                      03363
          param _ FALSE  param2 _ FALSE                               03364
          <"Infile?">                                                 03365
            ( "NO"                                                    03366
            / "YES" param _ LSEL( #"OLDFILELINK" )                    03367
            )                                                         03368
          <"Outfile?">                                                03369
            ( "NO"                                                    03370
            / "YES" param2 _ LSEL( #"NEWFILELINK" )                   03371
            )                                                         03372
        CONFIRM xxdetach( param, param2 );                            03373
    COMMAND %delete JNLDEL%                                           05108
      zxdelete =                                                      05109
        "DELETE"!L2! "JNLDEL"                                         05110
          <"from program buffer">                                     05111
          CONFIRM xjnldelete();                                       05112
    COMMAND %check results of running tasks%                          03374
      zxcheck =                                                       03375
        "CHECK" <"Results of Tasks">                                  03376
        CONFIRM xxcheck();                                            03377
    END.                                                              03144
% SYNTAX GENERATING SUBSYSTEM %                                      04158
  SUBSYSTEM subsyntax KEYWORD "SYNTAX GENERATOR"                      04159
    COMMAND zkshow =                                                  04160
      "SHOW"                                                          04161
      grammar _ NULL                                                  04162
                                                                      04163
        ( param _ "COMMAND" <"in Subsystem">                          04164
          param1 _ allsubs                                            04165
          param1 _ grammar                                            04166
          xkbuild( param1 )                                           04167
          syncur _ NULL                                               04168
          synsubs                                                     04169
          param1 _ syncur                                             04170
        / param _ "RULE"                                              04171
          param1 _ LSEL(#"WORD")                                      04172
        / param _ "SUBSYSTEM"                                         04173
          param1 _ allsubs                                            04174
          param1 _ grammar                                            04175
        )                                                             04176
      param2 _ #"ALL"                                                 04177
        %                                                             04215
        ( lookconfirm()                                               04178
        / param2 _                                                    04179
          ( "DNLS" <"Commands">                                       04180
          / "TNLS" <"Commands">                                       04181
```

```
            / "ALL" <"Commands"> )                                  04182
         )                                                          04183
         %                                                          04216
      CONFIRM                                                       04184
         xkshow( param, param1, param2);                            04185
   COMMAND zkcopy =                                                 04186
      "COPY"                                                        04187
      grammar _ NULL                                                04188
                                                                    04189
         ( param _ "COMMAND" <"in Subsystem">                       04190
            param1 _ allsubs                                        04191
            param1 _ grammar                                        04192
            xkbuild( param1 )                                       04193
            syncur _ NULL                                           04194
            synsubs                                                 04195
            param1 _ syncur                                         04196
         / param _ "RULE"                                           04197
            param1 _ LSEL(#"WORD")                                  04198
         / param _ "SUBSYSTEM"                                      04199
            param1 _ allsubs                                        04200
            param1 _ grammar                                        04201
         )                                                          04202
      <"to follow"> dest _ DSEL(#"STATEMENT")                       04203
      level _ LEVADJ                                                04204
      param2 _ #"ALL"                                               04205
         %                                                          04217
         ( lookconfirm()                                            04206
         / param2 _                                                 04207
            ( "DNLS" <"Commands">                                   04208
            / "TNLS" <"Commands">                                   04209
            / "ALL" <"Commands"> )                                  04210
         )                                                          04211
         %                                                          04218
      CONFIRM                                                       04212
         xkcopy( param, param1, param2, dest, level);               04213
   END.                                                             04214
FINISH OF NLSLANGUAGE                                               01249
% EXTERNAL KEYWORD DECLARATIONS MOVED HERE FOR SPLIT%               04446
   DECLARE EXT-KEYWORD % so only one copy exists in system These
   keywords are defined as external strings in FCONST. %              035
      % STRUCTURAL ENTITIES %                                        036
         "BRANCH",                                                  037
         "GROUP",                                                   038
         "PLEX",                                                    039
         "STATEMENT",                                               040
      % TEXTUAL ENTITIES %                                         041
         "CHARACTER",                                               042
         "INVISIBLE",                                               043
         "LINK",                                                    044
         "NUMBER",                                                  045
         "PASSWORD",                                               03149
         "TEXT",                                                    046
         "VISIBLE",                                                 047
         "WORD",                                                    048
      % MISC. ENTITIES %                                           049
         "FILE",                                                    050
```

```
        "OLDFILELINK",                                                  01527
        "NEWFILELINK",                                                  03000
        "NAME",                                                           051
        "RETURN",                                                       03025
        "FILERETURN",                                                   03026
        "IDENT",                                                        03639
        "IDENTLIST",                                                    03640
        "EDGE",                                                         03381
        "MARKER",                                                         053
        "REST",  % Used in HELP %                                       03983
        "BACK",  % Used in HELP %                                       03987
        "MENU", % Used in HELP %                                        03984
        "HLPCOM", % Used in HELP %                                      03985
        "CNTLQ"; % Used in HELP %                                       03986
FILE nlsident % CML <programs>identification.cml %% (CML,)
(programs,identification.cml,) %                                        03641
   % DECLARATIONS %                                                    03668
      DECLARE PARSEFUNCTION                                             03669
         answ,           % reads answer construct %                    03670
         answer,                % for questions - returns  0/1 %       03671
         sp,             % reads next char, TRUE if space %            03672
         readconfirm,    % reads next char if ca %                     03673
         readbug,        % reads next char if BUG %                    03674
         readoption,     % TRUE if next char is optchar %              03675
         readrepeat,     % TRUE if next char is repeat %               03676
         lookansw,       % TRUE if next char is Y/CA %                 03677
         lookconfirm,    % TRUE if next char is CA/REPEAT/INSERT %     03678
         lookbug,        % TRUE if next char is BUG %                  03679
         looknum,        % TRUE if next char is a number %             03680
         clearname,      % clears the name area %                      03681
         notca;          % reads next char, TRUE iff not CA char %     03682
      DECLARE EXT-KEYWORD % so only one copy exists in system These
      keywords are defined as external strings in CONST. %              03684
         % TEXTUAL ENTITIES %                                          03690
            "CHARACTER"!L2!,                                            03691
            "INVISIBLE"!L2!,                                            03692
            "LINK"!L2!,                                                 03693
            "NUMBER"!L2!,                                               03694
            "PASSWORD"!L2!,                                             03695
            "TEXT"!L2!,                                                 03696
            "VISIBLE"!L2!,                                              03697
            "WORD"!L2!,                                                 03698
         % MISC. ENTITIES %                                            03699
            "FILE"!L2!,                                                 03700
            "OLDFILELINK"!L2!,                                          03701
            "NEWFILELINK"!L2!,                                          03702
            "NAME"!L2!,                                                 03703
            "RETURN"!L2!,                                               03704
            "FILERETURN"!L2!,                                           03705
            "IDENT"!L2!,                                                03706
            "IDENTLIST"!L2!,                                            03707
            "EDGE"!L2!,                                                 03708
            "MARKER"!L2!;                                               03709
      DECLARE EXTERNAL % not defined here see (nls,pdata,) %            03710
         allsubs,                                                       03711
         nlssubs;                                                       03712
```

```
% IDENTIFICATION SUBSYSTEM COMMANDS %                                    02313
    SUBSYSTEM subident KEYWORD "IDENTIFICATION"                          02314
      COMMAND %verify  (record/ident-file)%                              02777
         ziverify =                                                      02778
            "VERIFY" <"Master Ident-File"> iwheel() param _
            #"EVERYTHING" param2 _ #"YES"                                02779
               [                                                         02781
                  (                                                      02782
                     param _ "INDIVIDUAL" <"branch"> /                   02783
                     param _ "GROUP" <"branch"> /                        02784
                     param _ "USED" <"idents"> /                         02785
                     param _ "ORGANIZATION" <"branch">/                  02786
                     param _ "EVERYTHING"/                               02787
                     "SINGLE" <"record for">                             02788
                        param _ LSEL(#"IDENT")                           02789
                     )                                                   02790
                  CLEAR                                                  03027
                  <"continue on errors?"> param2 _ answer()              02791
                  ]                                                      02792
               CONFIRM CLEAR <"type Control-O to abort, Control-S for
               status"> xiverfil(param, param2);                         02793
      COMMAND %update ident-file%                                        02758
         ziupdate =                                                      02759
            "UPDATE" <"master Ident-File"> anychanges()                  02760
            param _ TRUE [<"later"> param _ FALSE]                       02796
            CONFIRM xiupdate(param);                                     02797
      COMMAND %type%                                                     02406
         zitype =                                                        02407
            param _ "TYPE" <"of organization"> iwheel()                  02408
            param2 _ ( "USER" / "SERVER" / "TIP" / "INDEPENDENT" /
            "ASSOCIATE")                                                 02409
            CONFIRM setifield(param, param2);                            02410
      COMMAND %subcollections%                                           02403
         zisubcol =                                                      02404
            param _ "SUBCOLLECTIONS"!L2! zinewidlist;                    02405
      COMMAND %status of loaded record%                                  02399
         zistatus =                                                      02400
            "STATUS"!L2! <"of loaded record">                            02401
            CONFIRM xistatus(#"LOADED", #"ALL");                         02402
      COMMAND %show record%                                              02394
         zishow =                                                        02395
            "SHOW" "RECORD"  <"for ident"> param _ LSEL(#"IDENT")
            param2 _ #"ALL" [<"field:"> fieldnames]                      02396
            CONFIRM xistatus(param, param2);                             02397
         fieldnames =                                                    03028
            param2 _ "APPROVE" /param2 _ "CAPABILITIES"!L2! /param2 _
            "COORDINATOR" /param2 _ "COMMENTS"!L2! /param2 _
            "DELIVERY" <"modes"> /param2 _ "EXPAND" <"references">
            /param2 _ "FUNCTION" /param2 _ "GROUPS" /param2 _ "IDENT"
            / "NLS"!L2! <"mail"> ( "HOST" param2 _ #"NLSHOST" /
            "USER" <"name"> param2 _ #"NLSADD") / "NET"!L2! <"mail">
            ( "HOST" param2 _ #"NETHOST" / "USER" <"name"> param2 _
            #"NETADD") /param2 _ "HARDCOPY" <"mail address"> /param2
            _ "NAME" /param2 _ "MEMBERSHIP" /param2 _ "ORGANIZATION"
            /param2 _ "PHONE" /param2 _ "SECONDARY"!L2!
            <"organization"> /param2 _ "SUBCOLLECTIONS" /param2 _
```

```
      "TYPE" <"of organization">;                                    03029
   COMMAND %secondary organization%                                  02391
      zisecorg =                                                     02392
         param _ "SECONDARY"!L2! <"organization"> zinewtext;
                                                                     02393
   COMMAND %remove%                                                  02689
      ziremove =                                                     02690
         "REMOVE"                                                    02691
            ( param _ "RECORD" <"for"> iwheel() param2 _
            LSEL("IDENT"!L2!) /                                      02696
            param _ "FIELD" canmodify()                              02697
               ( param2 _ "COMMENTS" /                               02698
               param2 _ "FUNCTION" /                                 02702
               "MAIL"                                                02706
                  <"address for delivery type"> param2 _
                  newdelivery /                                      02714
               param2 _ "PHONE" /                                    02707
               param2 _ "SECONDARY"!L2! <"organization"> /           02708
               param2 _ "SUBCOLLECTIONS" ))                          02709
            CONFIRM                                                  02716
            xidelete (param, param2);                                02715
   COMMAND %phone%                                                   02388
      ziphone =                                                      02389
         param _ "PHONE" zinewtext;                                  02390
   COMMAND %organization%                                            02385
      ziorg =                                                        02386
         param _ "ORGANIZATION" zinewid;                             02387
   COMMAND %name%                                                    02368
      ziname =                                                       02369
         "NAME" canmodify() CONFIRM CLEAR                            02370
         <"old:"> shoifield(#"NAME")                                 02914
         ( isindividual()                                            02371
            CLEAR                                                    02372
            <"last name:"> param _ LSEL(#"TEXT")                     02373
               (isnewrec() CLEAR                                     02906
                  (checkname(param)                                  03146
                     <"is the desired person already in the Master
                     Ident-File?">                                   02838
                        (answ() abort()/ DUMMY)                      02907
                  / DUMMY )                                          03147
               / DUMMY )                                             03145
            CLEAR                                                    02374
            <"first name or initial:"> param2 _ LSEL(#"TEXT")
                                                                     02375
            CLEAR                                                    02376
            <"middle initial or name:"> param3 _ LSEL(#"TEXT")
                                                                     02377
         / CLEAR <"new:"> param _ LSEL(#"TEXT") param2 _ NULL
         param3 _ NULL )                                             02378
         setifield(#"NAME", param, param2, param3);                  02379
   COMMAND %membership%                                              02358
      zimembership =                                                 02359
         param _ "MEMBERSHIP" canmodify() <"old:">
         shoifield(param) PERFORM zinewmem UNTIL (<"finished?">
         answ());                                                    02360
      zinewmem =                                                     02361
```

```
        CLEAR                                                   02899
        ("ADD"  <"member(s)">                                   02362
           param2 _ LSEL(#"IDENTLIST") CONFIRM                  02363
           xiaddmem(param, param2) /                            02364
        "DELETE" <"member(s)">                                  02365
           param2 _ LSEL(#"IDENTLIST") CONFIRM                  02366
           xidelmem(param, param2));                            02367
     COMMAND %mail address%                                     02350
        zimail =                                                02351
           "MAIL"!L2! <"address type:"> canmodify() (           02352
              "HARDCOPY"                                        02353
                 <"old:"> shoifield(#"USADD") CLEAR <"new:"> param _
                 LSEL(#"TEXT") CONFIRM setifield(#"USADD", param) /
                                                                02917
              "NETWORK"                                         02946
                 param3 _ #"NETHOST" param4 _ #"NETADD" onlineadd /
                                                                02948
              "NLS"!L2!                                         02947
                 param3 _ #"NLSHOST" param4 _ #"NLSADD" onlineadd);
                                                                02949
        onlineadd =   %assumes param3 and param4 are set up%    02923
           <"old host:"> shoifiled (param3) CLEAR               02941
           <"new host:"> param2 _ LSEL(#"TEXT") CLEAR           02942
           <"old user name:"> shoifiled(param4)  CLEAR          02943
           <"new user name:"> param _ LSEL(#"TEXT")             02944
           CONFIRM setifield (param3, param2) setifield (param4,
           param);                                              02945
     COMMAND %load record%                                      02346
        ziload =                                                02347
           "LOAD" <"record for"> param _ LSEL(#"IDENT")         02348
           CONFIRM xiloadrecord(param);                         02349
     COMMAND %independent%                                      02905
        ziindependent =                                         02900
           param _ "INDEPENDENT"!L2! <"(not associated with an
           organization)"> canmodify() CONFIRM                  02901
           CLEAR  <"Hardcopy mail address:"> param _ LSEL(#"TEXT")
           CONFIRM                                              02902
           setifield(#"ORGANIZATION", #"INDEPENDENT")           02903
           setifield(#"USADD", param);                          02904
     COMMAND %ident%                                            02341
        ziident =                                               02342
           param _ "IDENT" (                                    02343
              isnewrec()                                        02884
                 (                                              02891
                    noident() CLEAR <"assigning tentative ident.
                    Please wait.">                              02894
                       asstentid() <"ident is"> shoifield(#"IDENT")
                                                                02878
                       CLEAR <"is this ident acceptable?">      02829
                          (answ() / PERFORM getid UNTIL
                          (okident(param)) )                    02830
                    / <"current ident:"> shoifield(#"IDENT")    02893
                       PERFORM getid UNTIL (okident(param))     02888
                  )                                             02892
              / iwheel() ipassword                              02883
                 <"current ident:"> shoifield(#"IDENT")         02896
```

```
                PERFORM getid UNTIL (okident(param))                02897
            )                                                       02885
         setifield(#"IDENT", param);                                02844
      getid =                                                       02845
         CLEAR <"type desired ident:"> param _ LSEL(#"TEXT");
                                                                    02898
      ipassword =                                                   02344
         <"password:"> xipassword(LSEL(#"PASSWORD"));               02345
   COMMAND %groups%                                                 02338
      zigroups =                                                    02339
         param _ "GROUPS"!L2! iwheel() canmodify() <"old:">
         shoifield(param) PERFORM zinewmem UNTIL (<"finished?">
         answ());                                                   02340
   COMMAND %function%                                               02334
      zifunction =                                                  02335
         param _ "FUNCTION" zinewtext;                              02336
   COMMAND %expand references%                                      02331
      ziexpand =                                                    02332
         param _ "EXPAND"!L2! canmodify() setifield(#"EXPAND",
         answer());                                                 02333
   COMMAND %delivery%                                               02324
      zidelivery =                                                  02325
         param _ "DELIVERY" canmodify()                             02326
            <"old:"> shoifield(param) CLEAR                         02327
            <"new:"> param2 _ newdelivery CONFIRM setifield(param,
            param2);                                                02328
      newdelivery =                                                 02329
         "NETWORK" / "NLS"!L2! / "HARDCOPY";                        02330
   COMMAND %coordinator%                                            02321
      zicord =                                                      02322
         param _ "COORDINATOR" zinewid;                             02323
   COMMAND %comments%                                               02318
      zimisc =                                                      02319
         param _ "COMMENTS"!L2! zinewtext;                          02320
   COMMAND %capabilities%                                           02315
      zicapabilities =                                              02316
         param _ "CAPABILITIES"!L2! iwheel() zinewtext;             02317
   COMMAND %approve record%                                         02411
      ziapprove =                                                   02412
         "APPROVE"!L2! <"loaded record"> iwheel()                   02413
         CONFIRM setifield(#"APPROVE", TRUE);                       02795
   COMMAND %add new record%                                         02380
      ziadd =                                                       02381
         "ADD" <"new record for a(n)">                              02382
         param _ ( "INDIVIDUAL" / "ORGANIZATION" / "GROUP")         02798
         CONFIRM newrec() setifield(#"RTYPE", param);               02834
   zinewtext = %assumes param set up%                               02426
      canmodify()  showold param2 _ #"TEXT" getnew                  03031
      CONFIRM setifield(param, param2);                             02429
   zinewidlist = %assumes param set up%                             02422
      canmodify()  showold param2 _ #"IDENTLIST" getnew             03032
      CONFIRM setifield(param, param2);                             02425
   zinewid = %assumes param set up%                                 02418
      canmodify() showold param2 _ #"IDENT" getnew                  03033
      CONFIRM setifield(param, param2);                             02421
   showold = CLEAR <"old:"> shoifield(param);                       02419
```

```
      getnew = CLEAR <"new:"> % (                                          03034
         readoption() <"use field"> fieldnames <"from record"> param2
         _ xistatus ( LSEL(#"IDENT"), param2) / %                          03044
         param2 _ LSEL(param2)%)%;                                         03043
      INITIALIZATION %load ident-file%                                     02414
         ziinit = xiloaidfil();                                            02415
      END.                                                                 02430
  FINISH                                                                   03644
% READMAIL SUBSYSTEM COMMANDS %                                           02688
   SUBSYSTEM subreadmail KEYWORD "READMAIL"                                02438
     COMMAND %verbose view%                                                02747
        zrverbose =                                                        02748
           "VERBOSE" <"view for catagory">                                 02619
              param _ LSEL(#"TEXT")                                        02620
              CONFIRM xjshwcat(param, #"VERBOSE") ;                        02621
     COMMAND %undelete%                                                    02749
        zrundelete =                                                       02750
           "UNDELETE"                                                      02614
              items                                                        02615
              <"in all catagories?">                                       02616
                 param2 _ answer()                                         02617
              CONFIRM xjdelitem(param, param2) ;                            02618
     COMMAND %show%                                                        02751
        zrshow =                                                           02752
           "SHOW"                                                          02611
              items CONFIRM xjshwitem (param) ;                            02612
                 %ALL is a keyword%                                        02613
     COMMAND %resend%                                                      02745
        zrresend =                                                         02746
           "RESEND"                                                        02606
              items                                                        02607
               <"when?"> param2 _ LSEL(#"TEXT") CONFIRM                    02608
                 %Will accept days of week or date%                        02609
              xjremind(param, param2) ;                                    02610
     COMMAND %output%                                                      02735
        zroutput =                                                         02736
           "OUTPUT"!L2!                                                    02741
              items                                                        02742
              <"to printer"> CONFIRM xjprinter(param) ;                    02743
                 %ALL will print whole catagory%                           02744
     COMMAND %omit%                                                        02737
        zromit =                                                           02738
           "OMIT"                                                          02592
              param2 _ NULL                                                02593
              param3 _                                                     02594
                 ("AUTHORS" param _ LSEL(#"TEXT") /                        02595
                    %ALL allowed%                                          02596
                 "TITLEWORDS" param _ LSEL(#"TEXT") /                      02597
                    %ALL allowed%                                          02598
                 "DATES" <"from"> param _ LSEL(#"TEXT") <"to"> param2 _
                 LSEL(#"TEXT") )                                           02599
                    %null date in from means nolimit, on to means
                    today%                                                 02600
              xjomit(param, param2, param3) ;                              02601
     COMMAND %next%                                                        02739
        zrnext =                                                           02740
```

```
      "NEXT"                                                          02590
         <"item"> CONFIRM xjshwitem(#"NEXT") ;                        02591
COMMAND %interrogate%                                                 02731
   zrinterrogate =                                                    02732
      "INTERROGATE"                                                   02583
         CONFIRM CLEAR                                                02584
         <"catagory:">                                                02585
            param _ LSEL(#"LINK")                                     02586
            xjsetcurcat(param)                                        02587
         PERFORM donext UNTIL (xjlast()) ;                            02588
            %xjlast returns TRUE if current item is last in
            catagory%                                                 02589
COMMAND %file%                                                        02721
   zrfile =                                                           02722
      "FILE"                                                          02575
         items <"under catagoy">                                      02576
         param2 _ LSEL("LINK"!L2!)                                    02577
         CONFIRM xjfileitem (param, param2) ;                         02578
COMMAND %expunge%                                                     02723
   zrexpunge =                                                        02724
      "EXPUNGE" <"all deleted items">                                 02573
         CONFIRM xjexpunge() ;                                        02574
COMMAND %delete%                                                      02725
   zrdelete =                                                         02726
      "DELETE"                                                        02566
         items                                                        02567
         <"from all catagories?">                                     02568
            param2 _ answer()                                         02569
         CONFIRM xjdelitem(param, param2) ;                           02570
COMMAND %catagory%                                                    02727
   zrcatagory =                                                       02728
      "CATAGORY"                                                      02562
         param _ LSEL(#"LINK")                                        02563
            %NEW is built in catagory (default).  what about
            AUTHOR!IDENT, CHRON, KEYWORDS!keyword, TITLE!WORD ??%
                                                                      02564
         CONFIRM xjsetcurcat(param) ;                                 02565
COMMAND %brief view%                                                  02719
   zrbrief =                                                          02720
      "BRIEF" <"view for catagory">                                   02559
         param _ LSEL(#"TEXT")                                        02560
         CONFIRM xjshwcat(param, #"BRIEF") ;                          02561
COMMAND %accept%                                                      02717
   zraccept =                                                         02718
      "ACCEPT"                                                        02549
         param2 _ NULL                                                02550
         param3 _                                                     02551
            ("AUTHORS" param _ LSEL(#"TEXT") /                        02552
               %ALL allowed%                                          02553
            "TITLEWORDS" param _ LSEL(#"TEXT") /                      02554
               %ALL allowed%                                          02555
            "DATES" <"from"> param _ LSEL(#"TEXT") <"to"> param2 _
            LSEL(#"TEXT") )                                           02556
               %null date in from means nolimit, on to means
               today%                                                 02557
         xjaccept(param, param2, param3) ;                            02558
```

```
    items =                                                          02623
      <"item number:"> param _ LSEL(#"TEXT");                        02624
    donext =                                                         02627
      param _ xjshwitem(#"NEXT")                                     02628
      <"file it?">                                                   02629
        (answ() <"catagory:">  param _ LSEL(#"LINK")
        xjfileitem(param, param2) /param _ NULL)                     02630
      <"forward it?">                                                02631
        (answ() <"to:"> param2 _ LSEL(#"TEXT") xjforward(param,
        param2) / param _ NULL)                                      02632
      <"delete it from this catagory?">                              02633
        (answ() xjdelitem(param) / param _ NULL);                     02634
    END.                                                             02648
% CALCULATOR SUBSYSTEM COMMANDS %                                   03988
  SUBSYSTEM subcalculator KEYWORD "CALCULATOR"                       03989
    INITIALIZATION                                                   03990
      zcinit=                                                        03991
        xcalcinit();                                                 03992
    TERMINATION                                                      03993
      zcterm=                                                        03994
        xcquit();                                                    03995
    RENTRY                                                           03996
      zclcrent =                                                     03997
        xcreenter();                                                 03998
                                                                     03999
    COMMAND zcshow =                                                 04000
      "SHOW"!L2!                                                     04001
      ("ACCUMULATOR"  <"Registers">                                  04002
        CONFIRM                                                      04003
        xcshoaccums()                                                04004
      /"FILE" ! NOTT!<"in window">                                   04005
        DSEL(#"EDGE")                                                04006
        CONFIRM                                                      04007
        xcshofil()                                                   04008
      );                                                             04009
    COMMAND zadd =                                                   04010
      ("ADD" / "+" )                                                 04011
        param2 _ LSEL(#"NUMBER")                                     04012
        CONFIRM                                                      04013
        %  execute ADD (param2)    %                                 04014
        xcarith(#"ADD",param2)                                       04015
        cfeedback();                                                 04016
    COMMAND zcclear =                                                04017
      "CLEAR"                                                        04018
        ("ACCUMULATOR"                                               04019
          CONFIRM                                                    04020
          xclraccum()                                                04021
          cfeedback()                                                04255
        /"FILE"                                                      04022
          CONFIRM                                                    04023
          xclrfil()                                                  04024
        );                                                           04025
    COMMAND zdivide =                                                04026
      ("DIVIDE" / "/" )                                              04027
        param2 _ LSEL(#"NUMBER")                                     04028
        CONFIRM                                                      04029
```

```
        %  execute DIVIDE (param2)   %                                  04030
        xcarith(#"DIVIDE",param2)                                       04031
        cfeedback();                                                    04032
                                                                        04033
   COMMAND zevaluate =                                                  04034
      "EVALUATE"!L2!                                                    04035
         param _ LSEL(#"TEXT")                                          04036
         xceval(param)                                                  04037
         (("ADD"  / "+" / CONFIRM)                                      04038
            param _ #"ADD"                                              04039
         /("DIVIDE"  / "/")                                             04040
            param _ #"DIVIDE"                                           04041
         /("MULTIPLY"  / "*"  / "X")                                    04042
            param _ #"MULTIPLY"                                         04043
         /("SUBTRACT"   / "-")                                          04044
            param _ #"SUBTRACT" )                                       04045
         CONFIRM                                                        04046
         xcevend(param)                                                 04047
         cfeedback();                                                   04048
   COMMAND zcformat =                                                   04049
      "FORMAT"                                                          04050
         ("PLACES" <"to the">                                           04051
            param _                                                     04052
            ("RIGHT"                                                    04053
            /"LEFT"                                                     04054
            )                                                           04055
            param2 _ LSEL(#"NUMBER")                                    04056
               CONFIRM                                                  04057
               xfdigits(param,param2)                                   04058
         /"COMMAS"                                                      04059
            param _ answer()                                            04060
            CONFIRM                                                     04061
            xcomma(param)                                               04062
         /param _                                                       04063
            ("LEFT"                                                     04064
            /"RIGHT")                                                   04065
            <"justify">                                                 04066
            CONFIRM                                                     04067
            xcjust(param)                                               04068
         /"TERSE"! NOTD! <"output">                                     04069
            param _ answer()                                            04070
            CONFIRM                                                     04071
            xcfeedb(param)                                              04072
         /"DOLLAR" <"signs">                                            04073
            param _ answer()                                            04074
            CONFIRM                                                     04075
            xdollar(param)                                              04076
         ) cfeedback();                                                 04077
   COMMAND zcinsert =                                                   04078
      "INSERT"                                                          04079
         <"accum following">                                            04080
         level _ NULL                                                   04081
         (ent _ ( text1 / "LINK" / "NUMBER" )                           04082
            dest _ DSEL(ent)                                            04083
         / ent _ "TEXT"                                                 04084
            dest _ DSEL(#"CHARACTER")                                   04085
```

```
        / ent _ structure                                                  04086
           dest _ DSEL(#"STATEMENT")                                       04087
           level _ LEVADJ)                                                 04088
        CONFIRM                                                            04089
        %  execute INSERT ACCUM (dest, param)    %                        04090
        xinsert(ent, dest, level,xcinsac());                               04091
   COMMAND zmultiply =                                                     04092
      ("MULTIPLY" / "*" / "X" )                                            04093
        param2 _ LSEL(#"NUMBER")                                           04094
        CONFIRM                                                            04095
        %  execute cmult (param2)   %                                      04096
        xcarith(#"MULTIPLY",param2)                                        04097
        cfeedback();                                                       04098
   COMMAND zsubtract =                                                     04099
      ("SUBTRACT" / "-" )                                                  04100
        param2 _ LSEL(#"NUMBER")                                           04101
        CONFIRM                                                            04102
        %  execute SUBTRACT (param2)   %                                   04103
        xcarith(#"SUBTRACT",param2)                                        04104
        cfeedback();                                                       04105
   COMMAND zcreplace =                                                     04106
      "REPLACE"                                                            04107
        dent _ editentity <"at">                                           04108
        dest _ DSEL (dent)                                                 04109
        sent _ #"TEXT"                                                     04110
        <"by accumulator">                                                 04111
        CONFIRM                                                            04112
        %  execute replace    %                                           04113
        xreplace(dent,dest,sent,xcinsac());                                04114
   COMMAND ztotal =                                                        04115
      "TOTAL"                                                              04116
        CONFIRM                                                            04117
        xctotl()                                                           04118
        ;                                                                  04119
   COMMAND zuse =                                                          04120
      "USE"                                                                04121
        ("ACCUMULATOR"  <"number">                                         04122
           param2 _ LSEL(#"NUMBER")                                        04123
           CONFIRM                                                         04124
           xcuseaccum(param2)                                              04125
        /"SAVED"  <"Accumulators">                                         04127
           CONFIRM                                                         04128
           xcusesaved()                                                    04129
        )cfeedback();                                                      04130
   COMMAND zwrite =                                                        04131
      "WRITE"  <"new">                                                     04132
        "FILE"                                                             04133
        param _ LSEL(#"NEWFILELINK")                                       04134
        CONFIRM                                                            04135
        xcwritef(param)                                                    04136
      ;                                                                    04137
   COMMAND znumber=                                                        04138
      looknum() %see if its another number %                               04139
      param2 _ LSEL(#"NUMBER")                                             04140
      CONFIRM                                                              04141
      xcarith(#"ADD",param2)                                               04142
```

```
    cfeedback();                                                    04143
                                                                    04144
END.                                                                04145
```