+INNLS

```
< NLS, FINTNLS.NLS.65, >, 23-Apr-78 22:59 JDH ;;;;
FILE fintnls  %  L10 to <rel-nls>FINTNLS %% (L10,) (rel-nls,FINTNLS,) %
                                                                      02
%global refs%                                                         03
   REF                                                                04
      tada, subda, cflda, namda, litda, msgda, echoda, ltvsda, vspcda;
                                                                      05
   REF rawchr, tda;                                                   06
%......Declarations......%                                           07
   REGISTER                                                           08
      a1 = 12, %working register%                                     09
      a4 = 15, %working register%                                    010
      r1 = 1,  %working register%                                    011
      r2 = 2,  %working register%                                    012
      r3 = 3,  %working register%                                    013
      r4 = 4,  %working register%                                    014
      r5 = 5,  %working register%                                    015
      rp = 11, %record pointer%                                      016
      p = 7,   %pattern stack%                                       017
      s = 9,   %general call stack pointer%                          018
      m = 10;  %mark stack pointer%                                  019
   EXTERNAL nls, start, init, gt, st, rentr;                         020
   DECLARE randm = 0, sequential = 1;                               01373
%.....Entry points.....%                                             021
   (dnls) PROCEDURE; %start display NLS%                             022
      nldevice _ devsri;                                             023
      nlparse _ TRUE;                                                024
      GOTO nls;                                                      025
      END.                                                           026
   (tnls) PROCEDURE; %start Typewriter NLS%                          027
      nldevice _ devtiex;                                            028
      nlparse _ TRUE;                                                029
      GOTO nls;                                                      030
      END.                                                           031
   (query) PROCEDURE; %start query system NLS%                       032
      exquery _ TRUE;                                                033
      nldevice _ devtiex;                                            034
      nlparse _ FALSE;                                               035
      GOTO nls;                                                      036
      END.                                                           037
%.....NLS initialization .....%                                      038
   (initnls) PROCEDURE; %start up the world%                         039
      % This is the procedure which is called in order to start up NLS%
                                                                     040
      %------------------%                                           041
      %!!!CANNOT HAVE LOCAL VARIABLES!!!%                            042
      (init): %label inserted to avoid the first instruction in the
      procedure which verifies that the stack pointer is reasonable%;
                                                                     043
      (nls): %pseudonym for init%;                                   044
      (start): %pseudonym for init%                                  045
      !JSP r1,l10init;  %set up l10 runtime environment%             046
      % go start the world %                                         047
         startup(); %does not return%                                048
      LOOP !haltf;  %just a precaution%                              049
      END.                                                           050
```

```
(frentr) PROCEDURE;                                                    051
   % come here on REENTER command %                                    052
   (rentr):                                                            053
   IF NOT continue THEN                                                054
      BEGIN                                                            055
      !JSP r1,l10init;  %initialize l10 runtime environment%           056
      typeas($"
      You must QUIT NLS before you can REENTER it!
      ");                                                              057
      LOOP !haltf;                                                     058
      END                                                              059
   ELSE GOTO init;                                                     060
   END.                                                                061
(startup) PROCEDURE; %initialize the world%                            062
   %get the device type, initials, and directory name of the user.
   Initializes the character input/output translation tables, using
   initch, initializes the rubout and control-C interupts, viewspecs,
   changes mode to work station for NLS, if not continuing.%           063
   %----------------%                                                  064
   LOCAL                                                               065
      fileno, da, fl, char, string, pcap, srr;                         066
      REF fl, da, srr;                                                 067
   LOCAL STRING gs[50], locstr[200];                                   068
   LOCAL TEXT POINTER p1, p2, p3, p4;                                  069
   % initialization done every startup or re-startup %                 070
      ON SIGNAL ELSE % Close files and issue halt jsys if error in
      startup %                                                        071
         BEGIN                                                         072
         IF sysmsg AND [sysmsg].M AND [sysmsg].L AND [sysmsg].M >=
         [sysmsg].L AND [sysmsg].L < 200 THEN typeas(sysmsg);          073
         (strthlt):                                                    074
            !JSYS 147B; %Reset jsys-- cannot use symbol because of
            conflict%                                                  075
             LOOP !haltf;                                              076
               %In case someone is foolish enough to try to contnue%
                                                                       077
         END;                                                          078
      % misc. stuff %                                                 079
         FIND SF(*arcstr*) ; %dummy successful FIND%                  01924
         lhostn _ hostnumber(); %save logical host number%             080
         tenex _ tenexverno();                                         082
         string _ 0;                                                   083
   IF continue THEN condevline(FALSE) %check for line number change,
   don't call initdis%                                                 01853
   ELSE % this stuff done only once %                                  084
      BEGIN                                                            085
      % get user & console number; and set some nice globals %        01374
         !gjinf;                                                       097
         console _ r4;                                                 098
         cuno _ r1;                                                    099
         nlmode _ typewriter; %in case needed for error msgs%          091
         &rawchr _ $getchar; %default lowest level input routine %
                                                                       092
         inpt _ $input;      %symbol conflict%                         093
         IF NOT autostrt THEN                                          094
            echon(); % turn wakeup on for all characters %             095
```

```
% setup user info, ident and user name string%                          0105
   IF SKIP !dirst($userstr + chbmty, cuno) THEN                         0106
      userstr.L _ slngth($userstr + chbmty, r1)                         0107
   ELSE *userstr* _ NULL;                                               0108
   IF t2v3flag THEN                                                    02138
      BEGIN                                                            02139
      *locstr* _ '<, *userstr*, '>, 0;                                 02145
      !rcdir(0, $locstr+chbmty);                                       02146
      logdirno _ (IF NOT r1 .A 6B10 THEN r3 ELSE 0);                   02147
      END                                                              02140
   ELSE                                                                02141
      BEGIN                                                            02142
      logdirno _ cuno;                                                 02144
      END;                                                             02143
   IF exquery THEN     % doesnt need an ident %                         0109
      BEGIN                                                            0110
      *initsr* _ "XXX";                                                0111
      char _ 'X-100B;                                                  0112
      cinit _ 0;                                                       0113
      cinit.cint1 _ char;                                              0114
      cinit.cint2 _ char;                                              0115
      cinit.cint3 _ char;                                              0116
      END                                                              0117
   ELSE                  % get the ident of user %                      0118
      BEGIN                                                            0119
      getuident(cuno);  %get user's ident%                             0120
      identwheel _ FALSE;                                              0121
      IF intmsf THEN %startup message -- intmsg string
      initialized in this file%                                        0122
         BEGIN                                                         0123
         !psout(chbmty + $intmsg);                                     0124
         crlf();                                                       0125
         IF nlmode = fulldisplay THEN                                  0126
            BEGIN                                                      0127
            typeas($intca);                                            0128
            LOOP                                                       0129
               BEGIN                                                   0130
               IF rawchr() = CA THEN EXIT LOOP;                        0132
               !pbout('?);                                             0133
               END;                                                    0134
            END;                                                       0135
         END;                                                          0136
      END;                                                             0137
% setup user-options or use system standards %                          086
   IF exquery THEN uoreset()                                          01889
   ELSE                                                               01890
      IF NOT uoget() THEN uoreset();                                    087
% now set user-options page to copy-on-write %                         088
   uoaccess($userdata / 1000B, racc .V cwacc);                          089
% terminal setup %                                                     090
   %get device%                                                        096
      IF exquery THEN setdev(nettty) ELSE                            01826
      getdev(console, nldevice);                                       0100
   % move to setdev when dcw does new tab stuff %                      0101
      IF nlmode = typewriter THEN setabs($stdtab);                     0102
% create timer fork %                                                  0103
```

```
      IF nlmode = fulldisplay THEN inittimer();                    0104
    % set character tables from user option page %                  0138
      initch (nldevice);                                           0139
    % Initialize interrupt system%                                  0140
      setinterrupts();                                             0141
    %get mode, system name%                                        0142
      *nlsnam* _ "NLS";                                            0147
      IF nlmode = typewriter THEN                                  0143
         nlstyp _ nttyp;                                           0148
      IF NOT nlparse THEN *nlsnam* _ "QUERY";                      0165
      nlssbn _ getsbn( $nlsnam );                                 01405
    % set viewspecs from options file %                             0166
      sysvspec _ stdvsp;                                           0167
      sysvspec[1] _ stdvsp[1];                                     0168
    % display area initialization%                                 0169
      dacnt _ 0;                                                   0170
      intdafl (&tda _ newda());                                    0171
         % initialize fields to standard values for a file window
         %                                                         0172
   END;                                                            0173
% allocate a block for the correspondence list %                  01955
   clistb _ getblk( (clistsz _ 200), $dspblk) + bhl;  %100
   entries%                                                       01956
% initialize display and work station
 or control character output control%                            0174
   initdis();                                                     0175
% startup back end %                                             0176
   startback(0);                                                  0177
% start or restart measurement%                                 01900
   meastart();                                                   01901
% get user's initial file or open after REENTER%                 0178
   IF NOT exquery THEN %only if not query from exec%             0179
      BEGIN                                                      0180
      IF NOT continue THEN                                       0181
         BEGIN                                                   0182
         %PROTOCOL%                                              0183
            curmkr _ 0;                                          0184
            curmkr.stfile _ dpyarea.dacsp.stfile _ fileno _
            openid();                                            0185
               curmkr.stpsid _ origin;                           0186
               curmkr[1] _ 1;                                    0187
            &fl _ flntadr(fileno);                               0188
            *locstr* _ NULL;                                    02149
            filnam( fileno, $locstr );                           0189
         pushfrring(dpyarea.dalink, $locstr, fileno);            0190
         readfrring(dpyarea.dalink, 0 : &srr);                   0191
         pushsrring(&srr, dpyarea.dacsp, dpyarea.daccnt,
         dpyarea.davspec, dpyarea.davspc2);                      0192
         %PROTOCOL%                                              0193
            % check for new journal mail%                        0194
               !gtfdb (fl.florig, 1000024B, $r3);                0195
               IF r3.ojdelf THEN %new mail%                      0196
                  BEGIN                                          0197
                  r2 _ 0; r2.ojdelf _ 1;                         0198
                  chnfdb (fl.florig, 24B, r2, 0);                0199
                  *gs* _ "You have new Journal mail";            0200
```

```
                    string _ $gs;                                   0201
                    END;                                            0202
                % do initial process commands %                     0203
                  CASE stupstr.L OF                                 0204
                    > 0:                                            0205
                      BEGIN                                         0206
                      ON SIGNAL ELSE EXIT CASE;                     0207
                      FIND SF(*stupstr*) ^p1 SE(*stupstr*) ^p2;     0208
                      p3 _ origin;                                  0209
                      p3.stfile _ dpyarea.dacsp.stfile;             0210
                      p3[1] _ 1;                                    0211
                      caddexp ($p1, $p2, $dpyarea, $p3 );           0212
                      IF p3 = endfil THEN EXIT CASE;                0213
                      p4 _ getend( p3 );                            0214
                      FIND SE(p4) ^p4;                              0215
                      auxstartup( $p3, $p4 );                       0216
                      END;                                          0217
                    ENDCASE;                                        0218
                  ON SIGNAL ELSE;                                   0219
                END                                                 0220
              ELSE                                                  0221
                %PROTOCOL%                                          0222
                  IF NOT openall() THEN                             0223
                    err($"Can't open any files");                   0224
              END;                                                  0225
        continue _ FALSE; %set true only by HALT%                   0226
        % transfer to start command parsing%                        0227
          IF nlmode NOT= typewriter THEN                            0228
            dstart(string)                                          0229
          ELSE tstart(string);                                      0230
        END.                                                        0231
                                                                    0232
%.....NLS pre-initialization and SSAVE code .....%                  0233
    SET lowsegtop = 120B, symtbltop = 116B;                         0234
    (saveit) PROCEDURE; %code used at save time%                    0235
      (st): % entry point to do partial initialization prior to ssave %
                                                                    0236
        bndchk _ FALSE;  % dont do boundary checks %                0237
        GOTO tt;                                                    0238
      (qt): % entry point to do partial initialization prior to ssave %
                                                                    0239
        bndchk _ TRUE;  % do boundary checks %                      0240
        GOTO tt;                                                    0241
      (tt): % entry point to do partial initialization prior to ssave %
                                                                    0242
        % set things up so can make calls %                        0243
          !JSP r1,l10init;                                          0244
        ON SIGNAL ELSE                                             01378
          BEGIN                                                    01379
          IF sysmsg AND [sysmsg].L <= [sysmsg].M AND [sysmsg].L IN [1,
          200] THEN                                                01389
            typeas(sysmsg);                                        01390
          !JSYS 147B; %Reset jsys-- cannot use symbol because of
          conflict%                                                01382
           LOOP !haltf;                                            01383
            %In case someone is foolish enough to try to contnue%
```

```
                                                                    01384
    END;                                                            01385
% set up rawchr for interaction %                                   01925
    &rawchr _ $getchar;                                             01926
% initialize character tables and terminal stuff%                   0247
    initch(devtiex);    % for query and general startup %           0248
    initbtbl(); % will be overlayed by useroptions data %           0249
    feedbk _ defdbk;                                                0250
    echon();                                                        0251
% check for loading page boundaries overflow %                      0252
    intmsf _ FALSE;  % no warning initially %                       0253
    IF bndchk THEN                                                  0254
       BEGIN                                                        0255
       IF $uopstart >= $userdata THEN                               0256
          BEGIN                                                     0257
          loderrm($"HIGH SEG LOADS IN USER-PROFILE PAGE");          0258
          intmsf _ TRUE;                                            0259
          END;                                                      0260
       IF ($enduop - $userdata) > 1000B THEN                        0261
          BEGIN                                                     0262
          loderrm($"USER-PROFILE PAGE TOO LONG");                   0263
          intmsf _ TRUE;                                            0264
          END;                                                      0265
       IF lowsegtop.LH >= symtbltop.RH THEN                         0266
          BEGIN                                                     0267
          loderrm($"CODE RUNS INTO SYMBOL TABLE");                  0268
          intmsf _ TRUE;                                            0269
          END                                                       0270
       ELSE                                                         0271
          IF (lowsegtop.LH + 2000B) >= symtbltop.RH THEN            0272
             BEGIN                                                  0273
             loderrm($"LESS THAN 2 PAGES FOR NEW SYMBOLS");         0274
             intmsf _ TRUE;                                         0275
             END;                                                   0276
       IF intmsf THEN *intmsg* _ CR, LF, "WARNING: SEE LOAD MAP
       FOR ERROR MESSAGES", 0;                                      0277
       END;                                                         0278
% set up shifts %                                                   0279
    cshift _ wshift _ nullch;  %no shift characters%                0280
    oshift _ FALSE;  %for output only -- do/dont print slashes
    for uppercase letters%                                          0281
% Set entry vector %                                               0282
    setvec();                                                       0283
% initialize free storage zone %                                   0284
    makezone($dspblk, 2000B, 8, $dspbke - $dspblk-1);               0285
% initialize system default viewspecs %                            0286
    defvs1 _ 0;                                                     0287
    defvs2 _ 0;                                                     0288
    defvs1.vstrnc _ defvs1.vslev _ 63;                              0289
    defvs1.vsindf _ defvs1.vsnamf _ defvs1.vsdaft _defvs1.vsnamf
    _ defvs1.vspagf _  TRUE;                                        0290
% host dependent%                                                  01894
    CASE (lhostn _ hostnumber()) OF                                 01906
       =isichost: oprtty _ arcoprtty;                               01907
       =isir1host: %NELC, was for ISID befor NOV 76%                01957
          %may want it for NELC%                                    01966
```

```
            BEGIN                                                    01961
            nojournalflag _ TRUE;                                    01965
            cpcldelim _ cpcrdelim _ '$;                              01962
            END;                                                     01964
         =isidhost, =isiehost: %may want to drop isid, depending
         on what isi does when isid goes to san diego%               01915
            BEGIN                                                    01916
            tops20flag _ TRUE;                                       01917
            devlproc _ 4; devsri _ 11;                               01960
            END;                                                     01918
         =sriai20host: %20-50%                                       01970
            BEGIN                                                    01974
            tops20flag _ TRUE;                                       01971
            devlproc _ 13;                                           01978
            imlac0_20B; imlac1_21B;                                  01980
            END;                                                     01973
         =sriaikahost:                                               01983
            BEGIN                                                    01984
            cpcldelim _ cpcrdelim _ '$ ;                             02136
            END;                                                     01985
         =office1host:                                               02000
            BEGIN                                                    02001
            cpcldelim _ cpcrdelim _ '$ ;                             02003
            END;                                                     02004
         =nsahost:                                                   01912
            BEGIN                                                    01913
            outremsiteflag _ inremsiteflag _ FALSE;                  01897
            *prtdir* _ "PRINTER";                                    01899
            oprtty _ nsaoprtty;                                      01977
            END;                                                     01914
         =utilhost: *subdir* _ "SUBSYS";                             01988
         ENDCASE                                                     01908
            BEGIN                                                    01920
            *opname* _ "<SUBSYS>NOUTPRC.SAV
            ";                                                       01919
            *subdir* _ "SUBSYS";                                     01990
            nojournalflag _ TRUE;                                    01922
            END;                                                     01921
      IF tops20flag THEN                                             01967
         BEGIN                                                       01968
         fvrdchar _ '. ;                                             01923
         t2fnsupdate($t2fsbegin+1, $t2fsend); %change semicolons
         to periods in system filenames%                             01946
         cpcldelim _ cpcrdelim _ '$;                                 01963
         END;                                                        01969
   % ask if there is a startup message %                             0293
      LOOP                                                           0294
         BEGIN                                                       0295
         typeas($"startup message?? ");                              0296
         CASE rawchr() OF                                            0298
            = 'y, = 'Y: %yes%                                        0299
               BEGIN                                                 0300
               typeas($"es.");                                       0301
               crlf();                                              01939
               gtintmsg();                                           0302
               EXIT LOOP;                                            0303
```

```
                END;                                                    0304
             = 'n, = 'N: %no%                                           0305
                BEGIN                                                   0306
                typeas($"o.");                                          0307
                crlf();                                                01940
                EXIT LOOP;                                              0308
                END;                                                    0309
             ENDCASE                                                    0310
                BEGIN                                                   0311
                typeas($" ? (y or n)");                                 0312
                crlf();                                                01941
                REPEAT LOOP;                                            0313
                END;                                                    0314
          END;                                                          0315
      %init backend% %take this out later when split complete%          0316
         bpresaveinit();                                                0317
      % save core image as disk file %                                  0318
         nlsave();                                                      0319
      % terminate partial initialization %                             0320
         LOOP !haltf;                                                   0321
   END.                                                                 0323
(loderrm)PROCEDURE(str);                                               01927
   %put out 4-line loading message%                                    01928
   REF str;                                                            01929
   crlf(); crlf();                                                     01930
   typeas($"************************************");                   01931
   crlf();                                                             01932
   typeas(&str);                                                       01933
   crlf();                                                             01934
   typeas($"************************************");                   01935
   crlf();                                                             01936
   RETURN;                                                             01937
   END.                                                                01938
(nlsave) PROCEDURE; % save nls as disk file %                           0325
   LOCAL jfn;                                                          0326
   LOOP % get save file name from user %                               0327
      BEGIN                                                            0328
      typeas($"save filename: ");                                      0329
      IF NOT SKIP !gtjfn(400003B6,100000101B) THEN                     0330
         BEGIN                                                         0331
         typeas($" ?");                                                0332
         crlf();                                                      01942
         REPEAT LOOP;                                                  0333
         END;                                                          0334
      jfn _ r1;                                                        0335
      EXIT LOOP;                                                       0336
      END;                                                             0337
   % save core image as dsk file %                                     0338
      !ssave(400000B6 .V jfn.RH, 777000B6 .V 520000B, 0);              0339
         % MODIFY BOUNDS LATER %                                       0340
   RETURN;                                                             0341
   END.                                                                0342
(t2insupdate) PROCEDURE(tabstrt, tabend);                              01943
   %change first semicolon in filename string to period for block of
   strings%                                                            01954
   LOCAL TEXT POINTER z1, z2;                                          01953
```

```
    REF tabstrt;                                                    01952
    WHILE tabstrt AND &tabstrt < tabend DO                          01947
       BEGIN                                                        01948
       IF FIND SF(*tabstrt*) [";] ^z2 ^z1 _z1 THEN ST z1 z2 _ ". ;
                                                                    01951
       &tabstrt _ &tabstrt + (tabstrt.M )/5 + 2;                    01950
       END;                                                         01949
    RETURN;                                                         01944
    END.                                                            01945
%.....Initialization support routines.....%                         0324
 (setvec)PROCEDURE;                                                 0343
    %Sets entry vector for frontend%                                0344
    entvec[0] _ $nls .V 254B9; %start NLS%                          0345
    entvec[1] _ $rentr .V 254B9; %reenter nls%                      0346
    entvec[2] _ $tnls .V 254B9; %start TNLS%                        0347
    entvec[5] _ $query .V 254B9; %start query%                      0349
    entvec[7] _ $dnls .V 254B9; %start DNLS%                        01881
    !sevec(4B5, 10B6 .V $entvec); %set entry vector%                0350
    RETURN;                                                         0351
    END.                                                            0352
 (intdafl) %initialize da record to standard values for a text area%
                                                                    0353
    PROCEDURE (da);                                                 0354
    LOCAL hinc, vinc, fv, index, cpos;                              0355
    REF da;                                                         0356
    hinc _ IF nlmode = typewriter THEN 1 ELSE tahinc;               0357
    vinc _ IF nlmode = typewriter THEN 1 ELSE tavinc;               0358
    %dislayarea record starting values%                             0359
       da.daseq _ FALSE;  %random display area%                     0360
       da.dafrozen _ FALSE;   %frozen boundary flag%                0361
       da.daauxiliary _ FALSE;   %for display purposes%             0362
       da.davspec _ stdvsp;                                         0363
       da.davspc2 _ stdvsp[1];                                      0364
       da.dalsrt _ 0; %so maklsrt will allocate space%              0365
       da.dacsp _ orgstid; %fileno later%                           0366
       da.daccnt _ 1;                                               0367
       da.dashrinkcnt _ 0;                                          0368
       da.dacsize _ tacsize;                                        0369
       da.danocs _ tacsize;                                         0370
       da.dasgcs _ tacsize;                                         0371
       da.daind _ indcnt * hinc;                                    0372
       da.damind _ tamind * da.daind;                               0373
       da.dalftjst _ 0;                                             0374
       da.damrow _ IF nlmode NOT= typewriter THEN                   0375
          ((tabottom-tatop)/vinc)*vinc ELSE linmax %lines to bottom
          margin%;                                                  0376
       %  Why did we ever do this?????                             01904
          IF nldevice = devlproc THEN da.damrow _ da.damrow - vinc;
                                                                    0377
       %                                                           01905
       da.damcol _ IF nlmode NOT= typewriter THEN                   01821
          IF udpcolmax > 0 AND udpwrapcol > 0 AND udpwrapcol <
          (taright - taleft)/hinc THEN                              01822
             udpcolmax                                              01823
          ELSE                                                      01824
             MIN(((taright-taleft)/hinc), IF udpcolmax=0 THEN 10000
```

```
            ELSE udpcolmax)*hinc                                    01825
        ELSE colmax * hinc %columns%;                               01820
        da.dahinc _ hinc;                                           0380
        da.davinc _ vinc;                                           0381
        da.danohi _ hinc;                                           0382
        da.dasghi _ hinc;                                           0383
        da.daleft _ taleft;                                         0384
        da.daright _ taright;                                       0385
        da.datop _ tatop;                                           0386
        da.dabottom _ tabottom;                                     0387
        da.daempty _ FALSE;                                         0388
        da.daexis _ TRUE;                                           0389
    %initialize display storage areas for LSRT and auxiliary LSRT%
                                                                    0390
        IF nlmode = fulldisplay THEN                                0391
            BEGIN                                                   0392
            aulsize _ da.damrow/da.davinc + 2; %number of useable lines
            + 1%                                                    0393
            IF NOT maklsrt ($dspblk, aulsize, &da)                  0394
            OR NOT (aulsrt _ getblk(aulsize * lsrtl, $dspblk)) THEN 0395
                err($"NLS out of display space");                   0396
            aulsrt _ aulsrt + bhl;  %first useable address of block%
                                                                    0397
            END;                                                    0398
    %last free lsrt "entry" - used for split screen size control% 0399
        rtfree _ MIN(rtmax, 60);                                    0400
    %fix up lsid bit table%                                         0401
        IF NOT da.dalsidb THEN                                      0402
            da.dalsidb _ ((&da - $dpyarea)/dal) * lsbtsize +
            $lsbitables;                                            0403
        dassnbit(da.dalsidb, -1, lsbtsize); %deassign all bits%     0404
    %set default tabstop values%                                    0405
        da.datab0 _ stdtab;                                         0406
        da.datab1 _ stdtab[1];                                      0407
        da.datab2 _ stdtab[2];                                      0408
        IF nlmode = typewriter AND NOT autostrt THEN setabs($stdtab);
                                                                    0409
    % set previous viewspec words %                                0410
        (da.dapvs, da.dapvs2) _ (da.davspec, da.davspc2);           0411
    RETURN;                                                         0412
    END.                                                            0413
                                                                    0414
(ckdvln)PROCEDURE; %if lp line number has changed, close and reopen
dspjfn%                                                             01829
    LOCAL lcontinue;                                                01868
    IF nldevice # devlproc THEN RETURN(FALSE);                      01830
    IF dspjfn # 0 THEN                                              01831
        BEGIN                                                       01832
        !gjinf();                                                   01833
        IF r4 = console THEN                                        01834
            BEGIN                                                   01870
            lcontinue _ continue := TRUE;                           01871
            getdev(console,nldevice); %restore device type lproc if it
            was changed%                                            01872
            continue _ lcontinue;                                   01873
            RETURN(FALSE);                                          01869
```

```
      END;                                                          01874
    console _ r4;                                                   01835
    IF NOT SKIP !closf(dspjfn) THEN NULL;                           01836
    getdev(console, nldevice);                                      01837
    END;                                                            01838
  RETURN(TRUE); END.                                                01839
(checkdevline) PROCEDURE; %if lp line number has changed, close and
reopen dspjfn and call initdis in "continue" mode%                  01840
  IF NOT ckdvln() THEN RETURN;                                      01841
  continue_TRUE;                                                    01842
  initdis();                                                        01843
  continue_FALSE;                                                   01844
  RETURN; END.                                                      01845
(condevline)PROCEDURE(initf); %check lp line number during a possible
"continue".  call initdis in "continue" mode if initf true. returns
true if it had to do something.%                                    01846
  LOCAL lcontinue, retflag;                                         01847
  lcontinue _ continue := FALSE;                                    01848
  IF retflag _ ckdvln() AND initf THEN                              01849
    BEGIN                                                           01854
    continue _ TRUE;                                                01856
    initdis();                                                      01857
    END;                                                            01855
  continue _ lcontinue;                                             01850
  RETURN(retflag); END.                                             01851
(getdev) PROCEDURE(lineno, device); %get user's device%             02005
  % this procedure uses the device in device if it is a valid device
  code (devsri for dnls, devtiex for tnls, or offline for dex),
  otherwise it uses the device info that it gets from the monitor%
                                                                    02006
  %----------------%                                               02007
  LOCAL devtype, i, char, count, code;                              02008
  !gttyp(lineno .V 4B5); %line number designator%                   02009
  devtype _ r2;                                                     02010
  IF device=devsri OR device=devlproc THEN                          02011
    CASE devtype OF                                                 02012
      =imlac0, =imlac1: NULL;                                       02013
        %if user has terminal type imlac, then believe him,
        otherwise believe device%                                   02014
      ENDCASE                                                       02015
        BEGIN %assume lineprocessor%                                02016
        devtype _ devlproc;                                         02017
        !sttyp(lineno .V 4B5, devtype);                             02018
        !rfmod();                                                   02019
        r2 _ r2 .A 777777777717B .V 4B10; %no raise, etc.%          02020
        !sfmod();                                                   02021
        !stpar();                                                   02022
        END;                                                        02023
  IF continue AND devtype = nldevice THEN RETURN;                   02024
  IF devtype = devlproc THEN                                        02025
    BEGIN                                                           02026
    IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN             02027
      err($"GTJFN failed, GETDEV");                                 02028
    dspjfn _ r1;                                                    02029
    IF NOT SKIP !openf(dspjfn, 100000385) THEN                      02030
      err($"OPENF failed, GETDEV");                                 02031
```

```
*altinp* _ NULL;                                                    02032
FOR i _ 0 UP UNTIL >= 10 DO                                         02033
   BEGIN                                                            02034
   % send and read big characters %                                 02035
      lpcmode();                                                    02036
   intter();                                                        02037
   !disms(1000);                                                    02038
   LOOP                                                             02039
      IF NOT SKIP !sibe(100B) THEN                                  02040
         BEGIN                                                      02041
         !bin( dspjfn );  *altinp* _ *altinp*, (char _ r2);
                                                                    02042
         IF char # lpresc THEN REPEAT LOOP;                         02043
         !bin( dspjfn );  *altinp* _ *altinp*, (code _ r2);
                                                                    02044
         IF code # irep THEN REPEAT LOOP;                           02045
         altinp.L _ altinp.L - 2;                                   02046
         lpxmax _ physxmax _ bincoord(dspjfn);                      02047
         lpymax _ bincoord(dspjfn);                                 02048
         lptype _ bincoord(dspjfn);                                 02049
         lpdlpad _ bincoord(dspjfn);                                02050
         lpbaudfactor _ bincoord(dspjfn);                           02051
         EXIT LOOP 2;                                               02052
         END                                                        02053
      ELSE REPEAT LOOP 2;                                           02054
   END;                                                             02055
IF i >= 10 THEN %interrogate has not been acknowldged yet%
                                                                    02056
   BEGIN                                                            02057
   typeas($"No Line-Processor data from INTERROGATE in
   GETDEV");                                                        02058
   LOOP !haltf;                                                     02059
   END;                                                             02060
litcolumn[0] _ 1;                                                   02061
!rfmod(100B);                                                       02062
r2 _                                                                02063
   r2 .A 740000777777B .V (lpymax * 2B8) .V (lpxmax * 1B6);
                                                                    02064
!sfmod( 100B, r2);                                                  02065
IF altinp.L THEN &rawchr _ $lpaltgetchar;                           02066
newcp _ scwindow _ FALSE;                                           02067
CASE lptype .A 4M OF                                                02068
   = datamedia:                                                     02069
      BEGIN                                                         02070
      putbackchar _ TRUE;                                           02071
      lpilpad _ 30;                                                 02072
      lpdlpad _ 1;                                                  02073
      END;                                                          02074
   = hazeltine: putbackchar _ TRUE;                                 02075
   =5: %datamedia 4000 possibly with CP and graphics%               02076
      BEGIN                                                         02132
      putbackchar _ newcp _ TRUE;                                   02077
      physxmax _ MIN(79, physxmax);                                 02134
      END;                                                          02133
   IN [6,15]: %datamedia 4000 with scroll window%                   02078
      %6 is only one assigned at present time, 7-15 default to
```

```
            these flags until they are assigned%                          02079
            BEGIN                                                         02080
            putbackchar _ newcp _ scwindow _ TRUE;                        02081
            physxmax _ MIN(79, physxmax);                                 02135
            END;                                                          02082
         ENDCASE putbackchar _ FALSE;                                     02083
      ldspjfn _ 0;  %no linked line-processors%                           02084
      END;                                                                02085
   IF devtype = imlac0 OR devtype = imlac1 THEN                           02086
      BEGIN                                                               02087
      IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN                 02088
         err($"GTJFN failed, GETDEV");                                    02089
      dspjfn _ r1;                                                        02090
      IF NOT SKIP !openf(dspjfn, 1000003B5) THEN                          02091
         err($"OPENF failed, GETDEV");                                    02092
      ldspjfn _ 0;  %no linked line-processors%                           02093
      END;                                                                02094
   CASE device OF                                                         02095
      = devsri: %user wants DNLS%                                         02096
         CASE devtype OF                                                  02097
            = imlac0, = imlac1, = devlproc: NULL;                         02098
            ENDCASE %not good%                                            02099
               BEGIN                                                      02100
               typeas($"not a display terminal");                         02101
               crlf();                                                    02102
               !JSYS haltf;                                               02103
               END;                                                       02104
      = devtiex: %user wants TNLS%                                        02105
         CASE devtype OF                                                  02106
            = imlac0, = imlac1, = devlproc:                               02107
               devtype _ devtiex;                                         02108
            ENDCASE;                                                      02109
      = offline:  %user wants DEX%                                        02110
         devtype _ offline;                                               02111
      ENDCASE; %use devtype%                                              02112
   CASE devtype OF                                                        02113
      =dev33, =dev35, =dev37, =devtiex, =imlac0, =imlac1, =nettty,
      =offline, = devlproc:                                               02114
         %defined device type%                                            02115
      ENDCASE devtype _ devtiex; %undefined, use devtiex%                 02116
   setdev(devtype);                                                       02117
   RETURN;                                                                02118
   END.                                                                   02119
                                                                          02120
(bincoord) PROCEDURE(jfn);                                                02121
   LOCAL char;                                                            02122
   !bin(jfn);                                                             02123
   IF r2#cooresc THEN RETURN(r2-40B);                                     02124
   char _ 0;                                                              02125
   !bin(jfn);                                                             02126
   char.xc1 _ r2 - 40B;                                                   02127
   !bin(jfn);                                                             02128
   char.xc2 _ r2 - 40B;                                                   02129
   RETURN(char);                                                          02130
   END.                                                                   02131
(setdev) PROCEDURE (device);                                               0490
```

```
%set up device dependent data for passed device%                    0491
LOCAL store, i, lmode;                                              0492
CASE nldevice _ device OF                                           0493
   = imlac0:     %vert tube, no LVH%                                0509
     BEGIN                                                          0510
     % normal char size:da 1; hinc: 9, 9, 9, 9; vinc: 30, 30, 30,
     30; grid: x: 0 to 719 by y: 0 to 719;%                         0511
     nlmode _ fulldisplay;                                          0512
     lmarg _ tmarg _ 0;                                             0513
     rmarg _ bmarg _ 719*256;                                       0514
     tacsize _ cflcsize _ namcsize _ subcsize _ litcsize _
     msgcsize _ ltcsize _ vscsize _ 1;                              0515
     tahinc  _ cflhinc _ namhinc _ subhinc _ lithinc _ msghinc _
     lthinc _ vshinc _ 9*256;                                       0516
     tavinc  _ cflvinc _ namvinc _ subvinc _ litvinc _ msgvinc _
     ltvinc _ vsvinc _ 18*256;                                      0517
     intdspda(tmarg + 7*tavinc, (bmarg/tavinc)*tavinc, lmarg,
     (rmarg/tahinc)*tahinc);                                        0518
        %initialize display areas for display terminal%             0519
     END;                                                           0520
  = imlac1:     %vert tube, LVH%                                    0521
     BEGIN                                                          0522
     % normal char size: 2; hinc: 16, 16, 16, 16; vinc: 16, 32,
     32, 48; grid: x: 0 to 1023 by y: 0 to 1023;%                   0523
     nlmode _ fulldisplay;                                          0524
     lmarg _ tmarg _ 0;                                             0525
     rmarg _ 1000*256;                                              0526
        %kludge because daccol field overflows in lsgfrmt%          0527
     bmarg _ 1023*256;                                              0528
     tacsize _ cflcsize _ namcsize _ subcsize _ litcsize _
     msgcsize _ 2;                                                  0529
     ltcsize _ vscsize _ 1;                                         0530
     tahinc  _ cflhinc _ namhinc _ subhinc _ lithinc _ msghinc _
     lthinc _ vshinc _ 16*256;                                      0531
     tavinc  _ cflvinc _ namvinc _ subvinc _ litvinc _ msgvinc _
     32*256;                                                        0532
     vsvinc _ ltvinc _ 16*256;                                      0533
     intdspda(tmarg + 7*tavinc, (bmarg/tavinc)*tavinc, lmarg,
     (rmarg/tahinc)*tahinc);                                        0534
        %initialize display areas for display terminal%             0535
     END;                                                           0536
  = devlproc:  %alpha-numeric display with line-processor%          0537
     BEGIN                                                          0538
     % char size: 1; hinc: 1; vinc: 1; grid: x: 0 to lpxmax by y:
     0 to lpymax;%                                                  0539
     nlmode _ fulldisplay;                                          0540
     tmarg _ lmarg _ 0;                                             0541
     rmarg _ lpxmax;                                                0542
     bmarg _ lpymax;                                                0543
     tacsize _ cflcsize _ namcsize _ subcsize _ litcsize _
     msgcsize _ ltcsize _ vscsize _ 1;                              0544
     tahinc  _ cflhinc _ namhinc _ subhinc _ lithinc _ msghinc _
     lthinc _ vshinc _ 1;                                           0545
     tavinc  _ cflvinc _ namvinc _ subvinc _ litvinc _ msgvinc _
     ltvinc _ vsvinc _ 1;                                           0546
     newintdspda(tmarg + 6*tavinc, bmarg, lmarg, rmarg);            0547
```

```
            %initialize display areas for display terminal%      0548
         IF &rawchr # $auxchr THEN                              01827
            &rawchr _ IF altinp.L THEN $lpaltgetchar ELSE $lpgetchar;
                                                                0549
         lppjfn _ 0; %make sure printer is off %                0550
         END;                                                   0551
      =dev33, =dev35, =dev37, =devtiex, =nettty, =offline:      0552
         BEGIN                                                  0553
         !rfmod(485 + console);                                 01882
         lmode _ r2;                                            01888
         IF lmode.fmodcol < colmax THEN                         01883
            BEGIN                                               01884
            lmode.fmodcol _ colmax;                             01885
            !stpar(r1, lmode);                                  01886
            END;                                                01887
         nlmode _ typewriter;                                   0554
         intdspda(0, pgsize, tpoffset, colmax);                 0555
         END;                                                   0556
      ENDCASE                                                   0557
         REPEAT CASE(nldevice _ devtiex);                       0558
   % initialize the translate tables %                          0559
      initch(nldevice);                                         0560
   % initialize the break table using "trnsli" %               0561
      initbtbl();                                               0562
   RETURN;                                                      0563
   END.                                                         0564
                                                                0565
(intdspda) PROCEDURE (tat, tab, tal, tar);  %set up display areas for
display terminal (relative to position of text area)%           0566
   %sets up top six lines as control ad literal feedback areas% 0567
   %text area%                                                  0568
      tatop _ tat;                                              0569
      tabottom _ tab;                                           0570
      taleft _ tal;                                             0571
      taright _ tar;                                            0572
   %Command Feedback Line%                                      0573
      cfltop _ tat - 4*tavinc;                                  0574
      cflbottom _ cfltop + 2*cflvinc;                           0575
      cflleft _ tal;                                            0576
      cflright _ tar;                                           0577
   %name%                                                       0578
      namtop _ cfltop;                                          0579
      nambottom _ namtop + namvinc;                             0580
      namleft _ cflleft;                                        0581
      namright _ cflright;                                      0582
   %lt viewspec%                                                0583
      lttop _ tat - 6*tavinc;                                   0584
      ltbottom _ lttop + ltvinc;                                0585
      ltleft _ tar - 15*lthinc;                                 0586
      ltright _ tar;                                            0587
   %viewspec%                                                   0588
      vstop _ ltbottom;                                         0589
      vsbottom _ vstop + vsvinc;                                0590
      vsleft _ ltleft;                                          0591
      vsright _ ltright;                                        0592
   %literal feedback%                                           0593
```

```
      lttop _ tatop - 2*tavinc;                                          0594
      litbottom _ tabottom;                                              0595
      litleft _ taleft;                                                  0596
      litright _ taright;                                                0597
   %message area%                                                        0598
      msgtop _ tat - 6*tavinc;                                           0599
      msgbottom _ msgtop + 2*tavinc;                                     0600
      msgleft _ cflleft + 13*msghinc;                                    0601
      msgright _ ltleft - msghinc;                                       0602
   %subsystem name%                                                      0603
      subtop _ tat - 6*tavinc;                                           0604
      subbottom _ subtop + subvinc;                                      0605
      subleft _ tal;                                                     0606
      subright _ msgleft - subhinc;                                      0607
   RETURN;                                                               0608
   END.                                                                  0609
(newintdspda) PROCEDURE (tat, tab, tal, tar);  %set up display areas
for display terminal (relative to position of text area)%                0610
   %sets up top six lines as control and literal feedback areas%         0611
   %text area%                                                           0612
      tatop _ tat;                                                       0613
      tabottom _ tab;                                                    0614
      taleft _ tal;                                                      0615
      taright _ tar;                                                     0616
   %other viewspecs area%                                                0617
      vstop _ tat - 4*tavinc;                                            0618
      vsbottom _ vstop + vsvinc;                                         0619
      vsright _ MIN(tar,physxmax);                                       0620
      vsleft _ vsright - 7*vshinc;                                       0621
   %lt viewspecs area%                                                   0622
      lttop _ tat - 4*tavinc;                                            0623
      ltbottom _ lttop + ltvinc;                                         0624
      ltright _ vsleft - tahinc;                                         0625
      ltleft _ ltright - 7*lthinc;                                       0626
   %sybsystem name area%                                                 0627
      subtop _ tat - 4*tavinc;                                           0628
      subbottom _ subtop + subvinc;                                      0629
      subleft _ tal;                                                     0630
      subright _ subleft + 14*subhinc;                                   0631
   %name area%                                                           0632
      namtop _ tat - 4*tavinc;                                           0633
      nambottom _ namtop + namvinc;                                      0634
      namleft _ subright + tahinc;                                       0635
      namright _ ltleft - tahinc;                                        0636
   %Command Feedback area%                                               0637
      cfltop _ tat - 3*tavinc;                                           0638
      cflbottom _ cfltop + 2*cflvinc;                                    0639
      cflleft _ tal;                                                     0640
      cflright _ tar;                                                    0641
   %literal feedback area%                                               0642
      lttop _ tatop - 2*tavinc;                                          0643
      litbottom _ tabottom;                                              0644
      litleft _ taleft;                                                  0645
      litright _ MIN(taright,physxmax);                                  0646
   %message (tty) area%                                                  0647
      msgtop _ tat - 6*tavinc;                                           0648
```

```
      msgbottom _ msgtop + 2*tavinc;                                    0649
      msgleft _ tal;                                                    0650
      msgright _ tar;                                                   0651
   RETURN;                                                              0652
   END.                                                                 0653
(initch) PROCEDURE (device); %initialize translation tables%            0654
   %Initialise the character set for the device%                        0655
   % legal device codes                                                 0656
      devsri, imlac0, imlac1, devtiex, dev33, dev35, dev37, offline,
      nettty%                                                           0657
   %--------------%                                                     0658
   LOCAL i;                                                             0659
   % 1st set tables to standard state %                                 0660
      FOR buffct _ 0 UP UNTIL = 128 DO                                  0661
         trnsli[buffct] _ trnslo[buffct] _ buffct;                      0662
   % Now set up control Characters %                                    0663
      %Literal  escape%                                                 0664
         todlit _ $ctlv;                                                0665
      %command accept%                                                 0666
         trnsli[4] %^D% _ CA;                                           0667
         trnslo[CA] _ nullch;                                           0668
      %command delete%                                                 0669
         trnsli[30B] _ CD; %^X%                                         0670
         trnsli[177B] _ CD; %rubout%                                    0671
         trnslo[CD] _ '#;                                               0672
      %center dot%                                                     0673
         trnsli[2B] _ C.; %^B%                                          0674
         trnslo[C.] _ '@;                                               0675
      %backspace character%                                            0676
         trnsli[1B] _ BC; %^A%                                          0677
         trnsli[10B] _ BC; %^H%                                         0678
         trnslo[BC] _ '\;                                               0679
      %backspace word%                                                 0680
         trnsli[27B] _ BW; %^W%                                         0681
         trnslo[BW] _ '_;                                               0682
      %backspace statement%                                            0683
         trnsli[13B] _ $ascbst; %^Q%                                    0684
         trnslo[$ascbst] _ '<;                                          0685
   % set up shifts and device dependent global stuff%                   0686
      CASE device OF                                                    0687
         =dev33, =dev35, =offline:                                      0688
            BEGIN                                                       0689
            %cause uppercase alpha's to get translated to lower case
            alpha's%                                                    0690
               FOR i _ 'A UP UNTIL > 'Z DO                              0691
                  trnsli[i] _ trnsli[i] + 40B;                          0692
            cshift _ '/;                                                0693
            wshift _ '\;                                                0694
            END;                                                        0695
         ENDCASE;                                                       0696
   % now do user-option stuff for this device %                        0697
      FOR i _ 0 UP UNTIL = 100 DO                                       0698
         BEGIN                                                          0699
         IF cctbl[i].ccdevice = device THEN                             0700
            BEGIN                                                       0701
            trnsli[(cctbl[i]).ccchar] _ cctbl[i].cctype;                0702
```

```
            trnslo[(cctbl[i]).cctype] _ cctbl[i].ccecho;          0703
            END;                                                  0704
         END;                                                     0705
   IF NOT libflg THEN BEGIN                                       0706
      r1 _ 18M;                                                   0707
      !JSYS rfmod;                                                0708
      IF r2 .A 10B %half duplex% THEN                             0709
         BEGIN %half duplex for net-tty%                          0710
         <NLS, INPFBK, echoff>();                                 0711
         feedbk _ 0;                                              0712
         END;                                                     0713
      END;                                                        0714
   RETURN;                                                        0715
   END.                                                           0716
                                                                  0717
(initdis) PROCEDURE;                                              0718
   LOCAL STRING send[10];                                         0719
   IF NOT autostrt THEN %set formatting for control characters%   0720
      BEGIN                                                       0721
      %set control character output codes%                        0722
      r1 _ 18M;                                                   0723
      IF nlmode = typewriter THEN                                01814
         BEGIN                                                   01815
         r2 _ IF spftab THEN ttycoc+2000B3                       01816
         ELSE ttycoc                                             01817
         END                                                     01818
      ELSE r2 _ dpycoc;                                          01819
      r3 _ 1B3;                                                   0725
      !JSYS sfcoc;                                                0726
      END;                                                        0727
   IF nlmode NOT= typewriter THEN %set up display areas%          0741
      setdis() %allocate or reallocate display areas%             0742
   ELSE %set lowercase if needed%                                 0743
      BEGIN                                                       0744
      CASE nldevice OF                                            0745
         =dev37, =nettty, =devtiex, =devlproc, =devsri, =imlac0,
         =imlac1:                                                 0746
            IF NOT autostrt THEN                                  0747
               BEGIN                                              0748
               !rfmod(100B);                                     01875
               r2 _ r2 .A 777777777717B .V 4B10;                  0751
               !sfmod();                                          0752
               !stpar(); %this one does "no raise"%              01876
               END;                                               0753
         ENDCASE;                                                 0754
      END;                                                        0755
   IF nlmode = fulldisplay THEN %clear screen and enable coordinate
   input%                                                         0728
      BEGIN                                                       0729
      %turn on special character input mode (allows for viewspec and
      marker input)%                                             01455
         lpcmode(); % enter coordinate mode %                    01459
      CASE nldevice OF %turn tty simulation off%                  0731
         = devlproc: NULL;                                        0732
         = imlac0, = imlac1:                                      0733
            BEGIN                                                 0734
```

```
            *send* _ begmsg, 1+remfudge, remtsf;                    0735
            !sout(dspjfn, chbmty+$send, -send.L);                   0736
            END;                                                    0737
          ENDCASE %tasker%                                          0738
            err($"No Tasker");                                      0739
      END;                                                          0740
   RETURN END.
                                                                    0756
(setdis) PROCEDURE; %set up display areas%                          0757
   %issue jsys's to allocate display area's for CFL, L-T area,
   viewspec area, name area,subsystem-name area, literal feedback
   area, and a message area.  Set up globals subda, cflda, ltvsda,
   vspcda, namda, litda, msgda, and assume dpyarea is set up for text
   area .%                                                          0758
   %------------------%                                            0759
   LOCAL  end, da, ls, save, consolenum, entryptr;                  0760
   LOCAL STRING send[10], str[40];                                  0761
   REF ls, da;                                                      0762
   cdtype _ dspno;                                                  0763
   %reset the display%                                              0764
      CASE nldevice OF                                              0765
         = devlproc:                                                0766
            BEGIN                                                  01401
            lprset();  %reset the display%                         01400
            cscreen(); %clear screen%                              01402
            lpcmode(); %go into coordinate input mode%             01403
            END;                                                   01404
         =imlac0, =imlac1: BEGIN                                    0767
            !sout(dspjfn, 4407B8+$rinistr, -11); %send 10 33B's plus
            one other char%                                         0768
            !disms(1000);                                           0769
            *send* _ begmsg, 1+remfudge, remlcm;                    0770
            !sout(dspjfn, chbmty+$send, -send.L);                   0771
            END;                                                    0772
         ENDCASE;                                                   0773
      dassnbit($dabt, -1, dabtsize);  %deassign all daid's%         0774
   % arm the cursor %                                              0775
      csrstr($rmdcr, tacsize, 0, 0);                                0776
   IF NOT continue THEN %must initialize%                           0777
      BEGIN                                                         0778
      &da _ $nlsdas;                                                0779
      %message area%                                                0780
         ttyda _ &msgda _ initda(&da, msgbase, msgx, msgy, $stn, 0,
         sequential);                                               0781
         ttysim _ msgda.dahandle;                                   0782
         ttywindow(ttyda);                                          0783
         &da _ &da + dal;                                           0784
      %command feedback area%                                       0785
         *cflstr* _ NULL;                                           0786
         *cflarw* _ "  ";                                           0787
         &cflda _ initda(&da, cflbase, cflx, cfly, $cflstr, $cflarw,
         randm);                                                    0788
         [cflda.dalsrt].rtx1 _ [cflda.dalsrt].rthinc;               0789
         &da _ &da + dal;                                           0790
      %L-T area%                                                    0791
         &ltvsda _ initda(&da, ltbase, ltx, lty, $stn, 0, randm);
```

```
                                                                    0792
      &da _ &da + dal;                                              0793
      IF nldevice = devlproc AND (lptype .A 4M) = deltadata THEN
                                                                    01397
         cline(0, ltvsda.datop, lpxmax);                            01398
            %This kludge is to get around timing problems on the
            delta data%                                             01399
   %view spec area%                                                 0794
      &vspcda _ initda(&da, vsbase, vsx, vsy, $stn, 0, randm);
                                                                    0795
      &da _ &da + dal;                                              0796
   %name area%                                                      0797
      &namda _ initda(&da, nambase, namx, namy, $stn, 0, randm);
                                                                    0798
      &da _ &da + dal;                                              0799
   %subsystem name area%                                            0800
      &subda _ initda(&da, subbase, subx, suby, $stn, 0, randm );
                                                                    0801
      &da _ &da + dal;                                              0802
   %literal display area%                                           0803
      &litda _ initda(&da,litbase, litx, lity, $lit, 0,
      sequential);                                                  0804
      litdahandle _ litda.dahandle;                                 0805
      litrmarg _ (litda.daright-litda.daleft)/litda.dahinc;         0806
      litreset _ TRUE;                                              0807
      litda.datab0 _ stdtab;                                        0808
      litda.datab1 _ stdtab[1];                                     0809
      litda.datab2 _ stdtab[2];                                     0810
      &da _ &da + dal;                                              0811
   %text area%                                                      0812
      &da _ $dpyarea;                                               0813
      UNTIL &da >= $dpyend DO                                       0814
         BEGIN                                                      0815
         IF da.daexis AND NOT da.dasuppress THEN                    0816
            <NLS, DSPGEN, alocda>(&da);                             0817
         &da _ &da + dal;                                           0818
         END;                                                       0819
   END                                                              0820
ELSE  %re-allocate them%                                            0821
   BEGIN                                                            0822
   IF &msgda THEN                                                   0823
      BEGIN  %old one has been released, get a new one%             0824
      <NLS, DSPGEN, alocda>(ttyda);                                 0825
      ttysim _ [ttyda].dahandle;                                    0826
      ttywindow(ttyda);                                             0827
      END;                                                          0828
   IF &namda THEN                                                   0829
      BEGIN                                                         0830
      &ls _ namda.dalsrt;                                           0831
      ls.rtlsid _ 0;                                                0832
      <NLS, DSPGEN, alocda>(&namda);                                0833
      dassnbit(namda.dalsidb, -1, lsbtsize);  %deassign all bits%
                                                                    0834
      IF dnstr.L THEN dn($dnstr);                                   0839
      END;                                                          0840
   IF &subda THEN                                                   0841
```

```
      BEGIN                                                          0842
      <NLS, DSPGEN, alocda>(&subda);                                 0843
      dassnbit(subda.dalsidb, -1, lsbtsize);  %deassign all bits%
                                                                     0844
      &ls _ subda.dalsrt;                                            0845
      ls.rtlsid _ 0;                                                 0846
      dsubsys($ssysname);                                            0847
      END;                                                           0848
   IF &cflda THEN                                                    0849
      BEGIN                                                          0850
      <NLS, DSPGEN, alocda>(&cflda);                                 0851
      dassnbit(cflda.dalsidb, -1, lsbtsize);  %deassign all bits%
                                                                     0852
      &ls _ cflda.dalsrt;                                            0853
      *cflstr* _ NULL;                                               0854
      ls.rtx2 _ ls.rtx1 + (cflstr.L - 1) * ls.rthinc;                0855
      ls.rtbps _ chbptr(empty) + $cflstr;                            0856
      ls.rtbpe _ chbptr(cflstr.L) + $cflstr;                         0857
      ls.rtlsid _ 0;                                                 0858
      &ls _ &ls + lsrtl;                                             0859
      *cflarw* _ "  ";                                               0860
      ls.rtbps _ chbptr(empty) + $cflarw;                            0861
      ls.rtbpe _ chbptr(cflarw.L) + $cflarw;                         0862
      ls.rtlsid _ 0;                                                 0863
      <NLS, DSPGEN, wrtstr>(&cflda, &ls);                            0864
      END;                                                           0865
   vspsav _ vspsav[1] _ 0;  %to make dspvsp work%                    0866
   IF &ltvsda THEN                                                   0867
      BEGIN                                                          0868
      <NLS, DSPGEN, alocda>(&ltvsda);                                0869
      dassnbit(ltvsda.dalsidb, -1, lsbtsize);  %deassign all bits%
                                                                     0870
      &ls _ ltvsda.dalsrt;                                           0871
      ls.rtlsid _ 0;                                                 0872
      IF &vspcda THEN                                                0873
         BEGIN                                                       0874
         <NLS, DSPGEN, alocda>(&vspcda);                             0875
         &ls _ vspcda.dalsrt;                                        0876
         ls.rtlsid _ 0;                                              0877
         IF nldevice = devlproc AND (lptype .A 4M) = deltadata
         THEN                                                        0878
            cline(0, ltvsda.datop, lpxmax);                          0879
               %This kludge is to get around timing problems on
               the delta data%                                       0880
         dspvsp(stdvsp, stdvsp[1], 3);                               0881
         END                                                         0882
      ELSE                                                           0883
         BEGIN                                                       0884
         IF nldevice = devlproc AND (lptype .A 4M) = deltadata
         THEN                                                        01391
            cline(0, ltvsda.datop, lpxmax);                          01392
               %This kludge is to get around timing problems on
               the delta data%                                       01393
         dspvsp(stdvsp, stdvsp[1], 1);                               0887
         END;                                                        0888
      END                                                            0889
```

```
      ELSE IF &vspcda THEN                                              0890
         BEGIN                                                          0891
         <NLS, DSPGEN, alocda>(&vspcda);                                0892
         dassnbit(vspcda.dalsidb, -1, lsbtsize);  %deassign all bits%
                                                                        0893
         &ls _ vspcda.dalsrt;                                           0894
         ls.rtlsid _ 0;                                                 0895
         IF nldevice = devlproc AND (lptype .A 4M) = deltadata THEN
                                                                        01394
            cline(0, ltvsda.datop, lpxmax);                             01395
               %This kludge is to get around timing problems on the
               delta data%                                              01396
         dspvsp(stdvsp, stdvsp[1], 2);                                  0898
         END;                                                           0899
      IF &litda THEN                                                    0900
         BEGIN                                                          0901
         litdahandle _ <NLS, DSPGEN, alocda>(&litda);                   0902
         dassnbit(litda.dalsidb, -1, lsbtsize);  %deassign all bits%
                                                                        0903
         litreset _ TRUE;                                               0904
         END;                                                           0905
      %text area%                                                       0906
         end _ (&da _ $dpyarea) + dal*dacnt;                            0907
         UNTIL &da >= end DO                                            0908
            BEGIN                                                       0909
            IF da.daexis AND NOT da.dasuppress THEN                     0910
               <NLS, DSPGEN, alocda>(&da);                              0911
               dassnbit(da.dalsidb, -1, lsbtsize);  %deassign all
               bits%                                                    0912
            &da _ &da + dal;                                            0913
            END;                                                        0914
      END;                                                              0915
   echoff();                                                            0916
   RETURN;                                                              0917
   END.                                                                 0918
                                                                        0919
                                                                        0920
(ttywindow) PROCEDURE (da);  %make a da a tty window%                   0921
   LOCAL STRING send[20];                                               0922
   REF da;                                                              0923
   CASE nldevice OF                                                     0924
      = devlproc:                                                       0925
         lpttywindow(da.datop, da.dabottom-da.davinc);                  0926
      = imlac0, = imlac1: %send out char string%                        0927
         BEGIN                                                          0928
         IF NOT da.dahandle THEN err($"Zero dahandle in ttywindow");
                                                                        0929
         *send* _ begmsg, 4+remfudge, echda, 50B, da.dahandle.daidr1
         + remfudge, da.dahandle.daidr2 + remfudge;                     0930
         !sout(dspjfn, chbmty+$send, -send.L);                          0931
         END;                                                           0932
      ENDCASE                                                           0933
         err($"No Tasker");                                             01804
   RETURN;                                                              0937
   END.                                                                 0938
(initda) PROCEDURE(da, base, x, y, str1, str2, type);                   0939
```

```
    LOCAL ls, end, nostrs;                                              0940
    REF str1, str2,  base, ls, da;                                      0941
    end _ &da + dal;                                                    0942
    DO da _ 0 UNTIL (&da _ &da + 1) >= end;                             0943
    &da _ end - dal;                                                    0944
    da.daccnt _ 0;                                                      0945
    da.dacsp _ endfil;                                                  0946
    da.dapstf _ -1;                                                     0947
    da.dacsize _ base.icsize;                                           0948
    da.dafont _ base.ifont;                                             0949
    da.dahinc _ base.ihinc;                                             0950
    da.davinc _ base.ivinc;                                             0951
    da.daleft _ base.ileft;                                             0952
    da.daright _ base.iright;                                           0953
    da.datop _ base.itop;                                               0954
    da.dabottom _ base.ibottom;                                         0955
    da.dalsz _ nostrs _ (base.ibottom-base.itop)/base.ivinc;            0956
    da.daseg _ type;                                                    0957
    da.daexis _ TRUE;                                                   0958
    da.dafrozen _ FALSE;                                                0959
    da.dalsidb _ ((&da-$nlsdas)/dal)*lsbtsize + $clsbitables;           0960
    dassnbit(da.dalsidb, -1, lsbtsize);  %deassign all bits%            0961
    alocda(&da);                                                        0962
    IF type = randm THEN %get lsrt for this display area%               0963
       BEGIN                                                            0964
       IF NOT (da.dalsrt  _ getblk (nostrs * lsrtl, $dspblk))THEN       0965
          err($"NLS error, out of display table space");                0966
       da.dalsrt _ da.dalsrt + bhl; %block header%                      0967
       &ls _ da.dalsrt;                                                 0968
       DO                                                               0969
          BEGIN                                                         0970
          ls.rtbps _ chbptr(empty) + &str1;                             0971
          ls.rtbpe _ chbptr(str1.L) + &str1;                            0972
          IF &str2 THEN &str1 _ &str2;                                  0973
          ls.rtcsize _ base.icsize;                                     0974
          ls.rtfont _ base.ifont;                                       0975
          ls.rthinc _ base.ihinc;                                       0976
          ls.rtx1 _ x;                                                  0977
          ls.rtx2 _ da.daright-da.daleft;                               0978
          ls.rty _ (y := y + base.ivinc);                               0979
          &ls _ &ls+lsrtl;                                              0980
          END                                                           0981
       UNTIL (nostrs _ nostrs -1) <= 0;                                 0982
       END;                                                             0983
    RETURN(&da);                                                        0984
    END.

                                                                        0985
(getuident) %get user's ident, given his login directory number in
USER%
PROCEDURE (user);                                                       01586
    % open group.index file and check user's ident.  If there is only
    one ident associated with this directory, use it.  If there are
    several, ask the user for his ident and check that it is in the
    list.  If there are no idents associated with this login
    directory, give error message and halt. %                           01587
     LOCAL i, jfn, chain, bp, idr2], wheelf, pcap, winit, idfrflag;
```

```
                                                                    01588
LOCAL STRING errorstr[200];                                         01589
user _ user.RH;                                                     02148
wheelf _ 0;                                                         01590
idfrflag _ nwheelf;                                                 01592
jfn _ 0;                                                            01593
ON SIGNAL ELSE                                                      01594
   BEGIN                                                            01595
   ON SIGNAL ELSE;                                                  01596
   IF jfn THEN sysclose(jfn);                                       01597
   jfn _ 0;                                                         01598
   END;                                                             01599
%read id from file or get from user%                                01600
   IF NOT idfrflag AND NOT (jfn _ sgtjfn(1B11, $grpfname ,
   $errorstr)) THEN                                                 01603
      idfrflag _ TRUE;                                              01604
   IF NOT idfrflag THEN                                             01601
      BEGIN                                                         01602
      IF NOT sysopen(jfn, rthawed, bintyp, $errorstr) THEN          01605
         err($"File associating login directories with NLS idents
         is bad.  Please report to ARC personnel.");                01606
      %get ident from group.index file or user%                     01607
         IF NOT SKIP !sfptr(jfn, user*4) THEN                       01608
            err($"File associating login directories with NLS
            idents is bad.  Please report to ARC personnel.");
                                                                    01609
         !bin(jfn); %get user directory number and group number%
                                                                    01610
         IF r2.LH NOT= user THEN                                    01611
            err($"File associating login directories with NLS
            idents is bad.  Please report to ARC personnel.");
                                                                    01612
         !bin(jfn); %get first word of ident or link to chain of
         idents%                                                    01613
      END;                                                          01614
   IF r2 = 0 OR idfrflag THEN                                       01615
      BEGIN                                                         01616
      IF wheelf OR idfrflag THEN idfrmuser ($initsr)                01617
      ELSE err($"no idents associated with this login directory");
                                                                    01618
      END                                                           01619
   ELSE IF r2.LH = 18M THEN %chain of idents%                       01620
      BEGIN                                                         01621
      chain _ r2.RH;                                                01622
      %get ident from user%                                         01623
         idfrmuser ($initsr);                                       01624
      LOOP                                                          01625
         BEGIN                                                      01626
         IF NOT SKIP !sfptr(jfn, chain) THEN                        01627
            err($"cant set position in file associating login
            directories an idents.  Please report to ARC
            personnel.");                                           01628
         !bin(jfn); %first word of ident%                           01629
         id _ r2;                                                   01630
         !bin(jfn); %second word of ident%                          01631
         id[1] _ r2;                                                01632
```

```
            IF initsr[1] = id AND initsr[2] = id[1] THEN                    01633
               EXIT LOOP; %found a match, everything is ok%                 01634
            !bin(jfn); %follow chain to next entry%                         01635
            IF (chain _ r2.RH) = 0 THEN %end of chain%                      01636
               BEGIN                                                        01637
               IF NOT wheelf THEN err($"ident not in list associated
               with this login directory");                                 01638
               typeas($"using ident not associated with this login
               directory");                                                 01639
               EXIT LOOP;                                                   01640
               END;                                                         01641
            END;                                                            01642
         END                                                                01643
      ELSE %get the ident%                                                  01644
         BEGIN                                                              01645
         initsr[1] _ r2;                                                    01646
         !bin(jfn);                                                         01647
         initsr[2] _ r2;                                                    01648
         bp _ chbmty + $initsr;                                             01649
         FOR i _ 1 UP UNTIL > 10 DO                                         01650
            IF ^bp = 0 THEN EXIT LOOP;                                      01651
         initsr.L _ i-1;                                                    01652
         END;                                                               01653
      CASE initsr.L OF                                                      01654
         > 4:                                                               01655
            err($"ident too long for use with NLS (max length is 4
            characters)");                                                  01656
         < 1: err($"Illegal ident.  Please report to ARC system
         personnel.");                                                      01657
         ENDCASE;                                                           01658
   IF jfn THEN sysclose(jfn);                                               01659
   jfn _ 0;                                                                 01660
   %now put initials into cinit, packed into 5-bit%                         01661
      cinit _ setcinit($initsr);                                            01662
   %see if pseudo wheel%                                                    01663
      i _ 0;                                                                01664
      WHILE NOT nwheelf AND (winit_nwheecinits[i:=i+1]) # 0 DO               01665
         IF winit=cinit THEN nwheelf_1;                                     01666
   RETURN;                                                                  01667
   END.

                                                                            01668
(setcinit) PROCEDURE (strng);                                               01061
   LOCAL count, char, val;                                                  01062
   REF strng;                                                               01063
   count _ val _ 0;                                                         01064
   UNTIL (count _ count + 1) > strng.L DO                                   01065
      BEGIN                                                                 01066
      CASE (char _ *strng*[count]) OF                                       01067
         IN ['A, 'Z]: char _ char-100B;                                     01068
         IN ['2, '6]: char _ char - 27B;                                    01069
         ENDCASE err($"Illegal character in id");                           01070
      CASE count OF                                                         01071
         =1: val.cint1 _ char;                                              01072
         =2: val.cint2 _ char;                                              01073
         =3: val.cint3 _ char;                                              01074
         ENDCASE val.cint4 _ char;                                          01075
```

```
      END;                                                          01076
    RETURN (val);                                                   01077
    END.

                                                                    01078
                                                                    01215
FINISH                                                              01216
This code can be deleted after 1-august-74.    -- Charles         01217
  (oldintdspda) PROCEDURE (tat, tab, tal, tar);  %set up display areas
  for display terminal (relative to position of text area)%         01218
    %sets up top six lines as control and literal feedback areas%
                                                                    01219
    %text area%                                                     01220
      tatop _ tat;                                                  01221
      tabottom _ tab;                                               01222
      taleft _ tal;                                                 01223
      taright _ tar;                                                01224
    %lt viewspec%                                                   01225
      lttop _ tat - 4*tavinc;                                       01226
      ltbottom _ lttop + ltvinc;                                    01227
      ltleft _ tal;                                                 01228
      ltright _ ltleft + 7*lthinc;                                  01229
    %viewspec%                                                      01230
      vstop _ tat - 3*tavinc;                                       01231
      vsbottom _ vstop + vsvinc;                                    01232
      vsleft _ tal;                                                 01233
      vsright _ vsleft + 7*vshinc;                                  01234
    %name%                                                          01235
      namtop _ tat - 4*tavinc;                                      01236
      nambottom _ namtop + namvinc;                                 01237
      namright _ tar;                                               01238
      namleft _ tar - 25*namhinc;                                   01239
    %sybsystem name%                                                01240
      subtop _ tat - 3*tavinc;                                      01241
      subbottom _ subtop + subvinc;                                 01242
      subright _ tar;                                               01243
      subleft _ subright - 25*subhinc;                              01244
    %Command Feedback Line%                                         01245
      cfltop _ tat - 4*tavinc;                                      01246
      cflbottom _ cfltop + 2*cflvinc;                               01247
      cflleft _ ltright + cflhinc;                                  01248
      cflright _ namleft - cflhinc;                                 01249
    %literal feedback%                                              01250
      littop _ tatop - 2*tavinc;                                    01251
      litbottom _ tabottom;                                         01252
      litleft _ taleft;                                             01253
      litright _ taright;                                           01254
    %message (tty) area%                                            01255
      msgtop _ tat - 6*tavinc;                                      01256
      msgbottom _ msgtop + 2*tavinc;                                01257
      msgleft _ tal;                                                01258
      msgright _ tar;                                               01259
    RETURN;                                                         01260
    END.                                                            01261
```

FONCY

```
< NLS, FONLY.NLS;6, >, 8-FEB-77 12:01 JDH ;;;;  % FRONT END ONLY
COMMANDS %
FILE fonly % L10 <rel-nls>fonly %%(110,) (rel-nls,fonly.rel,) %          02
%declarations%                                                           03
   REF rawchr, msgda, inpt, tda;                                         09
   REGISTER r1=1, r2=2, r3=3;                                          02364
%.....ROUTINES FROM THE EDITOR SUBSYSTEM %                             01879
   %clear%                                                             01880
      (xclear) %Execute Clear Command%                                 02548
         PROCEDURE                                                     02549
            %FORMALS%                                                  02550
               (result, %result record%                                02551
               parsemode);      %parsing, backup, cleanup%             02552
            LOCAL da, y, length; REF da, result;                       02553
         %----------------------%                                      02554
         CASE parsemode OF                                             02555
            = parsing:                                                 02556
               BEGIN                                                   02557
               &da _ ttyda;                                            02558
               length _ da.daright-da.daleft;                          02559
               FOR y _ da.datop UP da.davinc UNTIL = da.dabottom DO
                                                                       02560
                  cline(da.daleft, y, length);                         02561
               END;                                                    02562
            ENDCASE;                                                   02563
         RETURN(&result);                                              02564
         END.

   %connect%                                                           02565
      (xconnect) %Execute Connect Command%                             01905
         PROCEDURE                                                     02464
            %FORMALS%                                                  02465
               (result, %result record%                                02466
               parsemode,       %parsing, backup, cleanup%             02467
               entity,  %entity type%                                  02468
               destination,     %destination pointer%                  02469
               parameters);     %password or type of terminal
               connection%                                             02470
            LOCAL console, tp, dskcn1, dskcn2, dir1, dir2;             02471
            LOCAL STRING locstr[50], erstng[200], send[10];            02472
            REF                                                        02473
               result, entity, destination, parameters, tp;            02474
         %----------------------%                                      02475
         CASE parsemode OF                                             02476
            = parsing:                                                 02477
               CASE entity OF                                          02478
                  = 177 %- display -%:                                 02479
                     BEGIN                                             02480
                     &tp _ &destination+d2sel;                         02481
                     *locstr* _ destination tp;                        02482
                     console _ VALUE($locstr, 8);                      02483
                     %suppress the current display%                    02484
                        shutdis();                                     02485
                        CASE nldevice OF %turn tty simulation off%     02486
                           = devlproc: NULL;                           02487
                           = imlac0, = imlac1:                         02488
                                                                       02489
```

```
                    BEGIN                                          02490
                    *send* _ begmsg, 1+remfudge, remtsf;           02491
                    !sout(dspjfn, chbmty+$send, -send.L);
                                                                   02492
                    END;                                           02493
                  ENDCASE %tasker%                                 02494
                    err($"No Tasker");                             02495
      xconterm(console, IF parameters = 1 THEN TRUE ELSE
      FALSE  %input flag%);                                        02496
      LOOP inpt();                                                 02497
      END;                                                         02498
    = 47 %- tty -%:                                                02499
      BEGIN                                                        02500
      &tp _ &destination+d2sel;                                    02501
      *locstr* _ destination tp;                                   02502
      console _ VALUE($locstr, 8);                                 02503
      xconterm(console, IF parameters = 1 THEN TRUE ELSE
      FALSE %input flag%);                                         02504
      IF parameters = $input THEN                                  02505
         LOOP inpt();                                              02506
      END;                                                         02507
    = 9 %- directory -%:                                           02508
      BEGIN                                                        02509
      IF parameters = 0 THEN                                       02510
         BEGIN  % no password entered, ensure valid text
         pointers anyway %                                         02511
         FIND SF(*locstr*) ^parameters;                            02512
         parameters[ d2sel] _ parameters;                          02513
         parameters[ d2sel+1] _ parameters[1];                     02514
         END;                                                      02515
      !gjinf();                                                    02516
      dir1 _ r2;                                                   02517
      !gtdal(IF tops20flag THEN -1 ELSE 0);                        02518
      dskcn1 _ r2 - r1;                                            02519
      cconfildir(&destination, &destination+d2sel,
      &parameters, &parameters+d2sel);                             02520
      !gjinf();                                                    02521
      dir2 _ r2;                                                   02522
      !gtdal(IF tops20flag THEN -1 ELSE 0);                        02523
      dskcn2 _ r2 - r1;                                            02524
      IF dskcn1 > 0 THEN                                           02525
         BEGIN                                                     02526
         gdname( dir1, $locstr);                                   02527
         *erstng* _                                                02528
            EOL, *locstr*, " OVER ALLOCATION BY ",
            STRING(dskcn1), " PAGES!";                             02529
         END;                                                      02530
      IF (dskcn2 > 0) AND (dir1 # dir2) THEN                       02531
         BEGIN                                                     02532
         gdname( dir2, $locstr);                                   02533
         *erstng* _                                                02534
            *erstng*, EOL, *locstr*, " OVER ALLOCATION BY
            ", STRING(dskcn2), " PAGES!";                          02535
         END;                                                      02536
      IF erstng.L THEN dismes(2, $erstng);                         02537
      END;                                                         02538
```

```
            ENDCASE err(notyet);                                        02539
         ENDCASE;                                                       02540
      RETURN(&result);                                                  02541
      END.
                                                                        02542
   (xconterm)    %Connect TTY Command%                                  02443
      PROCEDURE(console, inout);                                        02444
      LOCAL i;                                                          02445
      linkcnsl _ TRUE;                                                  02598
      savnldevice _ nldevice;                                           02446
      %arm control-S to remove this connected state%                    02447
         savchntab[1] _ chntab[1] := $brkconnection .V 1B6;             02448
      %advise or link to designated terminal%                           02449
         IF inout THEN %advise%                                         02450
            BEGIN                                                       02451
            FOR i _ 20 DOWN UNTIL <= 0 DO                               02452
               BEGIN                                                    02544
               IF SKIP !adviz(2000004B5 .V console) THEN EXIT;          02545
               !disms(1000);                                            02546
               END;                                                     02547
            IF i <= 0 THEN                                              02454
               err($"That terminal did not issue Accept Connection
               command");                                               02455
            END                                                         02456
         ELSE %output linked only%                                      02457
            BEGIN                                                       02458
            IF  NOT SKIP !tlink(14B10 .V 777777B, console .V 400000B)
            THEN                                                        02459
               err($"Unable to connect terminals");                     02460
            END;                                                        02461
      RETURN;                                                           02462
      END.
                                                                        02463
%accept%                                                                02285
   (xaccept) %Execute Accept Connection Command%                        02368
      PROCEDURE                                                         02369
         %FORMALS%                                                      02370
            (result, %result record%                                    02371
            parsemode,       %parsing, backup, cleanup%                 02372
            inputf,           %input outpout or outputo only%           02373
            console);        %line number of other terminal%            02374
            REF                                                         02375
               result, console, inputf;                                 02376
         LOCAL i, tp, device, save;                                     02377
         LOCAL STRING locstr [10], send[10];                            02378
         REF tp;                                                        02379
         %----------------------%                                       02380
         CASE parsemode OF                                              02381
            = parsing:                                                  02382
               BEGIN                                                    02383
               device _ nldevice;                                       02384
               savnldevice _ nldevice;                                  02385
               %arm control-S to remove this connected state%           02386
                  savchntab[1] _ chntab[1] := $brkconnection .V 1B6;
                                                                        02387
               &tp _ &console+d2sel;                                    02388
```

```
*locstr* _ console tp;                                          02389
linkcnsl _ VALUE($locstr, 8);                                   02390
%set receive advise%                                            02391
   IF inputf=1 THEN                                             02392
      BEGIN                                                     02393
      FOR i _ 4 DOWN UNTIL <= 0 DO                              02394
         IF SKIP !adviz(1000004B5 .V linkcnsl) THEN EXIT
         LOOP;                                                  02395
      IF i <=  0 THEN                                           02396
         err ($"That terminal not trying to connect to
         you");                                                 02397
      !dismS(5000); %wait 5 sec for things to settle
      down%                                                     02543
      END;                                                      02398
IF nldevice = devlproc THEN                                     02399
   BEGIN                                                        02400
   save _ enablw();                                             02401
   *locstr* _ "TTY", console tp, ":, 0;                         02402
   IF NOT SKIP !gtjfn(1B6, ($locstr+1) .V 18M6) THEN
                                                                02403
      BEGIN                                                     02404
      linkcnsl _ 0;                                             02405
      err($"unable to accept connection -- cant get
      JFN");                                                    02406
      END;                                                      02407
   ldspjfn _ r1;                                                02408
   IF NOT SKIP !openf(ldspjfn, 100000385) THEN                  02409
      BEGIN                                                     02410
      reljfn(ldspjfn);                                          02411
      linkcnsl _ ldspjfn _ 0;                                   02412
      err($"unable to accept connection -- cant open tty
      in binary mode");                                         02413
      END;                                                      02414
   disablw(save);                                               02415
   END;                                                         02416
IF nlmode = fulldisplay THEN                                    02417
   BEGIN                                                        02418
   %shut down NLS display%                                      02419
      shutdis();                                                02420
      CASE nldevice OF %turn tty simulation off%                02421
         = devlproc: NULL;                                      02422
         = imlac0, = imlac1:                                    02423
            BEGIN                                               02424
            *send* _ begmsg, 1+remfudge, remtsf;                02425
            !sout(dspjfn, chbmty+$send, -send.L);               02426
            END;                                                02427
          ENDCASE %tasker%                                      02428
            err($"No Tasker");                                  02429
   %restore NLS display with proper formating and linked
   display areas%                                               02430
      %determine proper formatting%                             02431
         IF nldevice = devsri AND device NOT= devsri THEN
                                                                02432
            setdev(device);                                     02433
      continue _ TRUE;                                          02434
      <INTNLS, initdis>();                                      02435
```

```
                continue _ FALSE;                                    02436
                alldsp();                                            02437
              END;                                                   02438
           END;                                                      02439
        ENDCASE;                                                     02440
     RETURN(&result);                                                02441
     END.                                                            02442

%disconnect%                                                         01997
   (xdisconnect) %Execute Disconnect Command%                        01998
     PROCEDURE                                                       01999
        %FORMALS%                                                    02000
           (result, %result record%                                  02001
           parsemode);      %parsing, backup, or cleanup%            02002
           REF                                                       02003
              result;                                                02004
     %----------------------%                                        02005
     CASE parsemode OF                                               02006
        = parsing:                                                   02007
           rstconnection();                                          02008
        ENDCASE;                                                     02010
     RETURN(&result);                                                02011
     END.                                                            02012

%treeze%                                                             02013
   (xfreeze) %Execute Freeze Command%                                02014
     PROCEDURE                                                       02015
        %FORMALS%                                                    02016
           (result, %result record%                                  02017
           parsemode,       %parsing, backup, cleanup%               02018
           destination,     %destination pointer%                    02019
           vs);             %viewspec pointer%                       02020
           REF                                                       02021
              result, destination, vs;                               02022
        LOCAL da;  REF da; % ptr to display area %                   02023
     %----------------------%                                        02024
     CASE parsemode OF                                               02025
        = parsing:                                                   02026
           BEGIN                                                     02027
           &da _ lda();                                              02028
           IF da.davspec.vsfrzf THEN                                 02029
              dpset(dspstrc,destination,endfil,endfil)               02030
           ELSE dpset(dspno,endfil,endfil,endfil);                   02031
           xfresta (&da, destination, vs, vs[1]);                    02032
           END;                                                      02033
        ENDCASE;                                                     02034
     RETURN(&result);                                                02035
     END.                                                            02036

%release%                                                            02037
   (xrelease) %Execute Release Command%                              02038
     PROCEDURE                                                       02039
        %FORMALS%                                                    02040
           (result, %result record%                                  02041
           parsemode,       %parsing, backup, cleanup%               02042
           entity,  %entity type%                                    02043
```

```
            destination); %destination pointer%                          02044
            REF                                                          02045
               result, entity, destination;                              02046
         LOCAL da;  REF da;                                              02047
      %----------------------%                                           02048
      CASE parsemode OF                                                  02049
         = parsing:                                                      02050
            BEGIN                                                        02051
            &da _ lda(); %display area address%                          02052
            CASE entity OF                                               02053
               = 178 %- frozen -%:                                       02054
                  BEGIN                                                  02055
                  xrelsta (&da, destination);                            02056
                  END;                                                   02057
               = 95 %- all -%: %frozen statements%                       02058
                  xrelallsta(&da);                                       02059
               ENDCASE err(notyet);                                      02060
            %if frozen viewspec, recreate everything, else nothing%
                                                                         02061
               IF da.davspec.vsfrzf THEN                                 02062
                  dpset(dspallf,destination,endfil,endfil)               02063
               ELSE dpset(dspno,endfil,endfil,endfil);                   02064
            END;                                                         02065
         ENDCASE;                                                        02066
      RETURN(&result);                                                   02067
      END.

                                                                         02068
%simulate%                                                               02069
   (xsimulate) %Execute Simulate Command%                                02070
      PROCEDURE                                                          02071
         %FORMALS%                                                       02072
            (result, %result record%                                     02073
            parsemode,        %parsing, backup, cleanup%                 02074
            devicetype);      %devicetype type%                          02075
            REF                                                          02076
               result, filename, devicetype;                             02077
         LOCAL dev, modetype;                                            02078
      %----------------------%                                           02079
      CASE parsemode OF                                                  02080
         = parsing:                                                      02081
            BEGIN                                                        02082
            dev _ strdev(&devicetype :modetype);                         02083
            xsimdev(lda(), dev, modetype);                               02084
            cspupdate _ FALSE;                                           02085
            dpset(dspallf, endfil, endfil, endfil);                      02086
            END;                                                         02087
         ENDCASE;                                                        02088
      RETURN(&result);                                                   02089
      END.

                                                                         02090
%;%                                                                      02091
   (xcomment) PROCEDURE( % Execute comment command %                     02567
      % FORMAL ARGUMENTS %                                               02568
         resultptr,  % ptr to result record %                            02569
         parsemode); % interpreter parsing mode %                        02570
         REF resultptr;                                                  02571
```

```
      REF rawchr, msgda, inpt, tda;                                    02572
    LOCAL char;                                                        02573
    %----------%                                                       02574
    CASE parsemode OF                                                  02575
      = parsing:                                                       02576
        BEGIN                                                          02577
        cspupdate _ FALSE;                                             02578
        IF nldevice = devlproc THEN                                    02579
          BEGIN                                                        02580
          !bout( dspjfn, lpesc);                                       02581
          !bout( dspjfn, lpnocoor);                                    02582
          END;                                                         02583
        echolt();                                                      02584
        LOOP                                                           02585
          CASE char _ inpt() OF                                        02586
            = cachar, = CD:                                            02587
              BEGIN                                                    02588
              IF nldevice = devlproc THEN lpcmode();                   02589
              echoff();                                                02590
              RETURN (&resultptr);                                     02591
              END;                                                     02592
            ENDCASE;                                                   02593
        END;                                                           02594
      ENDCASE;                                                         02595
    RETURN (&resultptr);                                               02596
    END.
                                                                       02597
%.....JUMP COMMAND SUPPORT %                                           02115
  % MODIFY A BUG SELECT BY A DAE FOR JUMP %                            02116
    (xjdae) PROC(     % modify bug selection by a dae %                02117
    % FORMAL ARGUMENTS %                                              02118
      resultptr,  % ptr to the return record %                         02119
      pmode,      % parsing mode %                                      02120
      sourceptr); % ptr to the bug select record %                     02121
    % RETURNS %                                                       02122
      % word1-2, word3-4 text ptr to the modified selection %
                                                                       02123
    REF % VARIABLES %                                                  02124
      resultptr,                                                       02125
      sourceptr;                                                       02126
    %--------------------%                                             02127
    % process according to parsemode %                                02128
      CASE pmode OF                                                    02129
        = parsing:                                                     02130
          BEGIN                                                        02131
          cueflg _ FALSE;                                              02132
          resultptr _ sourceptr; resultptr[1] _ sourceptr[1];
                                                                       02133
          getaddr( &resultptr );                                       02134
          resultptr[2] _ resultptr; resultptr[3] _ resultptr[1];
                                                                       02135
          END;                                                         02136
      ENDCASE;                                                         02137
    RETURN (&resultptr);                                               02138
    END.
                                                                       02139
```

```
% DNLS jump support routines %                                          02140
   (xringjump) PROCEDURE( % provides feedback for stepping through
   the rings for jump RETURN and jump FILE RETURN %                     02141
      % In Help, the back message show ugly link syntax.%               02271
      % FORMAL ARGUMENTS %                                             02142
         resultptr,  % ptr to result record %                         02143
         parsemode,  % parsing mode %                                 02144
         entity);    % entity type for the jump %                      02145
      LOCAL da, stid, stdb, len, cc, pvs1, pvs2, srr, filestraddr;
                                                                       02146
      LOCAL TEXT POINTER tp1,tp2;                                      02147
      LOCAL STRING temp[20];                                           02148
      REF resultptr, entity, da, srr;                                  02149
      % -------------- %                                              02150
      CASE parsemode OF                                                02151
         = parsing:                                                    02152
            BEGIN                                                      02153
            %set da to current display area%                           02154
               &da _ lda();                                            02155
            CASE entity OF                                             02156
               = 37 %- return -%:                                      02157
                  BEGIN                                                02158
                  % advance through jump stack %                       02159
                     readfrring(da.dalink, 0 : &srr);                  02160
                     stid _ readsrring(&srr, srrcnt _ srrcnt+1 : cc,
                     pvs1, pvs2);                                      02161
                        %srrcnt is a global which is set to zero in
                        jrinit%                                        02162
                  %display the current statement%                      02163
                     % display first 20 chars of stmt in name area %
                                                                       02164
                        % get statement length %                      02165
                           IF NOT lodprop( stid, txttyp : stdb) THEN
                                                                       02366
                              err($"No text block associated with
                              node");                                  02367
                           len _ [stdb].schars + 1; % number of chars
                           %                                           02167
                        % construct text ptrs to each end of stmt %
                                                                       02168
                           tp1 _ stid;                                 02169
                           tp1[1] _ 1;                                 02170
                           tp2 _ stid;                                 02171
                           tp2[1] _ MIN(20,len); %truncate if over 20
                           chars %                                     02172
                        % assign something to the string "temp" %
                                                                       02173
                           IF len > 1 THEN *temp* _ tp1 tp2 ELSE
                           *temp* _ "<NULL>";                          02174
                     dn($temp); %display string%                       02175
                  % save the current state in the result record %
                                                                       02176
                     resultptr.retstid _ stid;                         02177
                     resultptr.retcc _ cc;                             02178
                     resultptr.retvs1 _ pvs1;                          02179
                     resultptr.retvs2 _ pvs2;                          02180
```

```
                END;                                                    02181
             = 36 %- filereturn -%:                                     02182
                BEGIN                                                   02183
                % move through the link stack one position,
                displaying the file name %                              02184
                   filestraddr _ readfrring(da.dalink, frrcnt _
                   frrcnt+1 : &srr);                                    02185
                      %frrcnt is a global which is set to zero by
                      jrinit%                                           02186
                   dn (filestraddr);                                    02187
                % save the values in the result record %               02188
                   resultptr.retstid _ readsrring(&srr, 0 : cc,
                   pvs1, pvs2);                                         02189
                   resultptr.retcc _ cc;                                02190
                   resultptr.retvs1 _ pvs1;                             02191
                   resultptr.retvs2 _ pvs2;                             02192
                   resultptr.retfn _ filestraddr;                       02193
                   resultptr.retsrr _ &srr;                             02194
                END;                                                    02195
             ENDCASE err( notyet );                                     02196
          END;                                                          02197
       = backup,                                                        02198
       = cleanup:                                                       02199
          dn( $""); % clear the name area %                             02200
       ENDCASE;                                                         02201
    RETURN( &resultptr );                                               02202
    END.
                                                                        02203
(jrinit) PROCEDURE( % initilize for jump return and jump file
return %                                                                02204
    % FORMAL ARGUMENTS %                                               02205
       resultptr,  % ptr to result record %                             02206
       parsemode,  % parsing mode %                                     02207
       entity);    %ptr to entity record%                               02208
    REF resultptr, entity;                                              02209
    % -------------- %                                                 02210
    CASE parsemode OF                                                   02211
       = parsing:                                                       02212
          BEGIN                                                         02213
          CASE entity OF                                                02214
             = 37 %- return -%: srrcnt _ 0;                             02215
             = 36 %- filereturn -%: frrcnt _ 0;                         02216
             ENDCASE err (notyet);                                      02217
          END;                                                          02218
       ENDCASE;                                                         02219
    RETURN( &resultptr );                                               02220
    END.
                                                                        02221
(xjmpcntdisp) PROCEDURE( % JUMP utility function which displays
the current contents of the content buffer (conreg) in the name
window %                                                                02222
    % FORMAL ARGUMENTS %                                               02223
       resultptr,  % ptr to the result record %                         02224
       parsemode); % parsing mode %                                     02225
    LOCAL tptr2;                                                        02226
    REF resultptr, tptr2;                                               02227
```

```
        % -------------- %                                                  02228
          CASE parsemode OF                                                 02229
             = parsing:                                                     02230
                BEGIN                                                       02231
                &tptr2 _ &resultptr + d2sel;                                02232
                FIND SF(*conreg*) ^resultptr SE(*conreg*) ^tptr2;
                                                                            02233
                dn( $conreg );                                              02234
                END;                                                        02235
             = backup,                                                      02236
             = cleanup:                                                     02237
                dn( $"" );                                                  02238
             ENDCASE;                                                       02239
          RETURN (&resultptr );                                             02240
          END.
                                                                            02241
FINISH of frontend                                                          01748
```

FRECORDS

```
< NLS, FRECORDS.NLS;5, >, 16-JUL-76 07:18 JDH ;;;;
FILE frecords % L10  <REL-NLS>frecords %% (L10,) (rel-nls,frecords.rel,)
%                                                                    0214
   (initialvalues) RECORD                                            0215
      ileft[36],                                                     0216
      iright[36],                                                    0247
      itop[36],                                                      0246
      ibottom[36],                                                   0245
      icsize[36],                                                    0243
      ifont[36],                                                     0244
      ihinc[36],                                                     0242
      ivinc[36];                                                     0241
% records for parser, select, pdata, psupport, etc. %                 02
   % for building ? string %                                         03
      (fbhlprec) RECORD                                              04
         hbfor[18],     % forward pointer to next string %           05
         hbbck[18],     % back pointer to previous string %          06
         hbval[36],     % value of this string %                     07
         hbastr[36];    % adr of this string %                       08
            SET EXTERNAL                                             09
               fbhlpl= 3;                                           0240
   % SUBSYSTEM DISPATCH RECORD %                                     010
      (sdispatch) RECORD % *** do not change this record without
      changing the CML compiler which outputs this record *** %      011
         dptname[36],   % ptr to subsystem name string %            012
         dptrun[18],    % ptr to commands rule / 0 %                 013
         dptvalid[18],  % dispatch record validation code %         014
         dptinit[18],   % ptr to initialization rule / 0 %          015
         dptfinish[18], % ptr to termination rule / 0 %             016
         dptnotused[18],        % not yet used %                    017
         dptrentry[18]; % ptr to rentry rule / 0 %                  018
   % VIEWSPEC RETURN RECORD %                                       019
      (pvsrecord) RECORD                                             020
         vs1[36],       % updated viewspec word 1 %                 021
         vs2[36],       % updated viewspec word 2 %                 022
         vscacode[18],  % ptr to user content analyzer pgm %        023
         vsusqcod[18];  % ptr to user sequence gen. pgm %           024
      % a viewspec collection astring follows the pvsrecord in the
      function state record %                                        025
   % LEVADJ RETURN RECORD %                                         026
      (plvrecord) RECORD                                             027
         plvcount[36];  % resolved level adjust count %            028
                        % u= +1, d= -1, etc %                       029
      % a level adjust collection astring follows the plvrecord in the
      function state record %                                        030
   % SELECTION RETURN RECORD %                                      031
      (pselrecord) RECORD                                            032
         sltp11[36],    % text ptr to head of selection %           033
         sltp12[36],                                                034
         sltp21[36],    % text ptr to tail of selection %           035
         sltp22[36];                                                036
   % CONFIRM RETURN RECORD %                                        037
      (pconrecord) RECORD                                            038
         pcomcode[36];  % command completion code %                 039
   % KEYWORD RECOGNITION RETURN RECORD %                            040
      (pkeyrecord) RECORD                                            041
```

```
      pkstrptr[36];  % ptr to global keyword string %                  042
% SBSTACK ENTRY DEFINITIONS %                                          043
   (sbentry) RECORD                                                    044
      sbptr[18],      % ptr to subsystem grammar %                     045
      sbnptr[18],     % ptr to subsystem name %                        046
      sbcount[18],    % execution bound count %                        047
      sbmode[4],      % subsystem processing mode %                    048
      sbpmode[4];     % previous subsystem processing mode %           049
% SELECTION STATE RECORD %                                            050
   (selstate) RECORD                                                   051
      entype[18],     % selection entity type %                        052
      sel1fun[18],    % ptr to first processing function %             053
      sel2fun[18],    % ptr to second processing function %            054
      cfl1len[7],     % initial length of cfl buffer %                 055
      cfl2len[7],     % length of CFL buffer after 1ST sel %           056
      nselects[2],    % number of selctions processed %                057
      maxselect[2],   % max # selects allowed %                        058
      litptr[18],     % ptr to literal string in any %                 059
      dolitreset[1]; % true if need to reset lit %                    060
% DEFINES LAYOUT OF WORK AREA FOR SAVING STATE %                       061
   (staterec) RECORD                                                   062
      currcm[9],      % current recognition mode %                     063
      optcount[7],    % $option nesting depth %                        064
      endopt[1],      % end of optional list flag %                    065
      optflag[1],     % only optional keywords acceptable %           066
      firstinst[18], % ptr to first inst. in text %                   067
      currinst[18],  % ptr to current inst. in text %                 068
      currpath[18],  % ptr to current path stack record%              069
      savstk[36],     % saved value of ptrstk entry on hit%           070
      savex[6],       % ptrx value when hit occurs %                  071
      firstx[ 6],     % initial index in ptrstk %                     072
      nextx[ 6],      % next index in ptrstk %                        073
      usrmode[6];     % current user recognition mode %               074
% DEFINES FUNCTION STATE RECORD %                                     075
   (retrec) RECORD                                                     076
      % the following four lines represent a quick and dirty
      temporary change to nls to fix the following problem. The
      result record is being used by keyword recognizer to store
      state information accessed via this record. However the parse
      functions, notably xselect already use this area to stoore
      their state information. The insertion of the four dummy words
      in the record will hopefuly temorarily avoid the issue. %   0235
         retrc1[36], %dummy word used to offset record%           0236
         retrc2[36], %dummy word used to offset record%           0237
         retrc3[36], %dummy word used to offset record%           0238
         retrc4[36], %dummy word used to offset record%           0239
      retval[36],     % return value %                               077
      inpsav[9],      % starting index for inpt buffer %             078
      fblen[9],       % initial length of feedback buffer %          079
      keyinlength[9],         % current length of keyinpstr %        080
      ksaveflag[9];  % current value of keysaveflag %               081
% DEFINES MODE BITS IN COMMAND MODE FIELDS %                          082
   (moderec) RECORD                                                    083
      tnlscmd[1],     % available in TNLS %                           084
      dnlscmd[1],     % available in DNLS %                           085
      l1cmd[1];       % level 1 command (single char recognition) %
```

```
                                                                        086
% INTERPRETER INSTRUCTION FORMATS %                                     087
   (instformat) RECORD                                                  088
      alternative[18],% address of alternative instruction %            089
      nsuccessor[18],        % address of nsuccessor instruction %
                                                                        090
      addr[18],      % address/value field %                           091
      val2[9],       % val2/ second value %                           092
      ctrl[3],       % control bits %                                  093
      opcode[6];     % operation code %                                094
% CHARACTER ACTION RECORDS %                                           095
   (actions) RECORD % defines actions associated with the recognition
   of a particular character %                                          096
      propcode[18],% opcode of process instruction %                    097
      insptr[18],    % ptr to first inst in grammar for this branch %
                                                                        098
      fbstrptr[18];  % ptr to prompt string associated with function
      %                                                                 099
% FUNCTION STATE RECORD %                                             0100
   (funstaterec) RECORD                                               0101
      a1word[36],    % first argument return word %                    0102
      a2word[36],    % second arg. return word %                       0103
      a3word[36],    % etc. %                                         0104
      a4word[36],                                                      0105
      t1word[36],    % temporary storage for use by function          0106
                     to record state %                                 0107
      t2word[36],    % ditto, etc. %                                  0108
      t3word[36],                                                      0109
      t4word[36],                                                      0110
      t5word[36];                                                      0111
% PATH STACK RECORD %                                                 0112
   (pathsr) RECORD                                                    0113
      pfunction[18], %addr of processing function/ NULL%              0114
      begnodeptr[18],        %nodeptr when starting to process node%
                                                                       0115
      curnodeptr[18],        %nodeptr currently in use%               0116
      markptr[18],   % ptr to last selection path %                   0117
      argcount[3],   % number of arguments to the function %         0118
      pmode[3],      % parsing mode %                                 0119
      ptrxsav[5],    % initial ptrx value %                           0120
      evalxsav[5],   % initial evalx value %                         0121
      evalmod[2],    % eval mode after mode was evaluted %            0122
      fcounter[10];  % frame counter %                               0123
%Record Definitions%                                                 0124
   (adar1)      %allocate display area record for r1%                0125
      RECORD                                                         0126
      adauly[10],   %upper left y-cordinate%                         0127
      adaulx[10],   %upper left x- cordinate%                        0128
      adasize[6],   %max no. of strings allowed in da%               0129
      adaseq[1];    % TRUE: sequential display area, FALSE: random%
                                                                     0130
   (adar2)      %allocate display area record for r2%               0131
      RECORD                                                         0132
      adalry[10],   %lower right y-cordinate%                        0133
      adalrx[10],   %lower right x- cordinate%                       0134
      adadcs[2],    %default character size%                         0135
```

```
   adadhi[6],    %default horizontal increment%                          0136
   adadf[5];     %default font%
                                                                         0137
(ccdefine) RECORD % format of entries in table "cctbl" %                 0138
   ccdevice[7],       % internal device code %                           0139
   cctype[7],         % cntrlchar %                                      0140
   ccchar[7],         % cntrlchar psuedonym %                            0141
   ccecho[7];         % to be echoed as %                                0142
(cords) RECORD                                                           0143
   xcord[18],  %x-cordinate for cursor word%                             0144
   ycord[18]; %y-cordinate for cursor word%
                                                                         0145
(daidr)   RECORD  %daid record for remote displays%                     0146
   daidr2[6],    %low order 5 bits%                                      0147
   daidr1[6];    %high order five bits%                                  0148
                                                                         0149
(formatr)  RECORD  %format byte in STRDA command to remote displays%
                                                                         0150
   xyds[1],    %TRUE: use old coordinates, FALSE read coordinates
   from command string%                                                  0151
   sdd[1],    %TRUE: use default da character size%                      0152
   sds[1],    %TRUE: use old string character size%                      0153
      %sdd=FALSE AND sds=FALSE: read character size from command
      string%                                                            0154
   idd[1],    %TRUE: use default da horizontal increment%                0155
   ids[1],    %TRUE: use old string horizontal increment%                0156
      %idd=FALSE AND ids=FALSE: read horizontal increment from
      command string%                                                    0157
   fdd[1],    %TRUE: use default da font%                                0158
   fds[1];    %TRUE: use old string font%                                0159
      %fdd=FALSE AND fds=FALSE: read font from command string%           0160
                                                                         0161
(frozen) RECORD                                                          0162
   fzvspec[36],                                                          0163
   fzvspc2[36],                                                          0252
   fzstid[36],                                                           0251
   fznext[18],                                                           0250
   fzexis[1];
                                                                         0249
   DECLARE EXTERNAL                                                      0164
      frzl= 4;
                                                                         0248
(jstatesave) RECORD % saves display area jump state info into
destination record %                                                     0165
   retstid[36],       % stid of new target from return ring %           0166
   retfn[18],         % address of file name string %                   0167
   retcc[12],         % char count of new target %                      0168
   retvs1[36],        % viewspecs -- word 1 %                           0169
   retvs2[36],        % viewspecs -- word 2 %                           0170
   retsrr[36];        % statement return ring address %                 0171
(prrecord) RECORD % saves parse rule info for REPLACE PARSERULE
COMMAND %                                                                0182
   prwrd1[36],        % first word of replaced CML inst. %              0183
   prwrd2[36],        % second word of replaced CML inst. %             0184
   praddr[18],        % address of replaced instruction %               0185
   prexists[1];       % TRUE if entry is still valid %                  0186
```

```
(scsrr) RECORD    %set cursor record for r3%                          0187
   sccsiz[2],     %character size%                                    0188
   schinc[6],     %horizontal increment%                              0189
   scfont[5];     %font%
                                                                      0190
(strda1) RECORD   %string manipulation record for r1%                 0191
   sstrid[18],    %string identirier%                                 0192
   sdaid[18];     %display area identifier%
                                                                      0193
(strda4) RECORD   %string manipulation record for r4%                 0194
   sycord[10],    %y-cordinate%                                       0195
   sxcord[10],    %x- cordinate%                                      0196
   scsize[3],     %character size%                                    0197
   shinc[7],      %horizontal increment%                              0198
   sfont[6];      %font%
                                                                      0199
(termtype) RECORD %terminal type (for use with strmt and rtrmt
jsies)%                                                               0200
   trmtyp[6],    %terminal type code: lpdy, pdpy, ltty%              0201
   grdtyp[4],    %grid type%                                         0202
   notused[9],                                                       0203
   ttylno[18];   %line number (-1= controlling tty)%                 0204
                                                                      0257
(fmoderec) RECORD %file mode word (for use with rfmod, sfmod, and
stpar jsiesZ)%                                                        0253
   fmodright[18],  %replace this with more detailed breakdown as
   needed%                                                            0255
   fmodcol[7];   %columns%                                           0254
                                                                      0256
(xc) RECORD    %xcord record for remote displays%                     0205
   xc2[6],     %low order 5 bits%                                    0206
   xc1[6];     %high order five bits%                                0207
                                                                      0208
(yc) RECORD    %ycord record for remote displays%                     0209
   yc2[6],     %low order 5 bits%                                    0210
   yc1[6];     %high order five bits%                                0211
                                                                      0212
% Measurement records %                                              0263
(msfjrec) RECORD %ffjobltab word format%                              0264
   msfltime[20],    %login time rel syst. startup in seconds%        0265
   msfmflag[1],   %flag this job being measured%                     0266
   msfmtyp[3],    %measurement type%                                 0267
   msffno[9];     %output file count (for file name%                 0268
(msrrec) RECORD %msrtab word format%                                  0269
   msrr0[9],                                                         0270
   msrr1[9],                                                         0271
   msrr2[9],                                                         0272
   msrr3[9];                                                         0273
DECLARE EXTERNAL                                                      0274
   msrtl=6,    %number of words in msrtab%                           0275
   msrmod=4;   %number of fields in msrtab word%                     0276
% Record definitions for HELP %                                      0217
(contrg) RECORD % a context ring item-- two words long %              0218
   constd[36], % stid of principal node of this context %            0219
   constk[18], % pointer to top of menu stack for this context %     0220
                %It will be zero if unused %                         0221
```

```
   totcnt[15], % total number of items in the stack %                    0222
   stktyp[3]; % type of stack-- menu, multiple, unused %
                                                                        0223
(stkhd) RECORD % 0th element of context stack block %                   0224
   conlnk[36], % link to next block in this stack; -1 if unlinked, 0
   if a free block %                                                    0225
   stkcnt[5]; % number of elements in this block (<= 25) %
                                                                        0226
(stkitm) RECORD % normal context stack elements; use word refereneces
rather than these fields to speed up.  Just here for information %
                                                                        0227
   mnstd[36], % stid of this stack item %                               0228
   stkhsh[36]; % name hash of this item; -1 if no name. %
                                                                        0229
(quspecs) RECORD %Query directives embedded in data base%               0230
   querinc[1],     %Include substructure at the node in the menu.%
                                                                        0231
   quercol[1],     %During printout, "columnate" all menu items.%
                                                                        0232
   querprt[1],     %Print QUERY viewspecs and links for debugging%
                                                                        0233
   queropts[8];    %Sets the number of options printed at one time.%
                                                                        0234
FINISH                                                                  0213
```

FRONTEND

```
< NLS, FRONTEND.NLS.18, >, 23-Feb-78 15:09 JDH ;;;;  % FRONT END SUPPORT
CODE %
FILE frontend % L10 <rel-nls>frontend %%(110,) (rel-nls,frontend.rel,) %
                                                                        02
%declarations%                                                          03
   DECLARE EXTERNAL                                                     04
       tbound = 0, bbound = 1, lbound = 2, rbound = 3, rtlast = 180;    05
   DECLARE EXTERNAL                                                     06
       hinc0 = 10, hinc1 = 14, hinc2 = 18, hinc3 = 22, vinc0 = 25, vinc1
      = 30, vinc2 = 35, vinc3 = 45;                                     07
   REGISTER r1 = 1, r2 = 2;                                             08
   REF rawchr, msgda, inpt, tda;                                        09
%.....files in display areas.....%                                     010
   (frefint)  PROCEDURE;                                               011
      %free and close files for files in file status table             012
      that are't referenced in display table dacsp or frozen            013
      list%                                                             014
      %----------------%                                               015
      LOCAL fileno, used, fl, da, endfl, endda;                         016
      REF fl, da;                                                       017
      IF filcnt NOT IN [0, filmax] OR dacnt NOT IN [1, damax]           018
      THEN                                                              019
         err($"NLS system error");                                      020
      endfl _ (&fl _ $filst) + filcnt*filstl;                           021
      endda _ $dpyarea + dacnt * dal;                                   022
      fileno _ 1;                                                       023
      UNTIL &fl >= endfl DO                                             024
         BEGIN                                                          025
         IF NOT fl.flnoclos THEN                                        026
            IF fl.flexis THEN                                           027
               BEGIN                                                    028
               &da _ $dpyarea;                                          029
               used _ FALSE;                                            030
               UNTIL &da >= endda DO                                    031
                  BEGIN                                                 032
                  IF filusd(fileno, &da) THEN                           033
                     BEGIN                                              034
                     used _ TRUE;                                       035
                     EXIT;                                              036
                     END;                                               037
                  &da _ &da + dal;                                      038
                  END;                                                  039
               IF NOT used THEN                                         040
                  BEGIN                                                 041
                  %                                                     042
                  &da _ $dpyarea;                                       043
                  UNTIL &da >= endda DO                                 044
                     BEGIN                                              045
                     &frr _ da.dalink;                                  046
                     ednfrr _ &frr + frrhlen + (frrelen*frr.frhlast);
                                                                        047
                     FOR &fre _ &frr + frrhlen UP frrelen UNTIL > endfrr
                     DO                                                 048
                        IF [fre.frring].srhexis AND
                        [fre.frring].srhfileno = fileno THEN            049
                           [fre.frring].srhfileno _ 0;                  050
```

```
                    END;                                                    051
                  %                                                         052
                  close(fileno);                                            053
                  END;                                                      054
               END;                                                         055
         BUMP fileno;                                                       056
         &fl _ &fl + filstl;                                                057
         END;                                                               058
      RETURN;                                                               059
      END.                                                                  060

   (tilusd)  PROCEDURE (fileno, da);                                        061
      %Given a file number and the address of a display area, this
      routine returns TRUE if the file is used in the frozen list or csp
      associated with the display area; otherwise it returns FALSE.%    062
      %-----------------%                                                   063
      LOCAL fl, fz;                                                         064
      REF fl, fz, da;                                                       065
      IF da.daempty OR da.dacsp = endfil THEN RETURN(FALSE);                066
      IF fileno = da.dacsp.stfile THEN RETURN(TRUE);                        067
      &fz _ da.dafrzl;                                                      068
      WHILE &fz DO                                                          069
         BEGIN                                                              070
         IF fz.fzexis AND fz.fzstid.stfile = fileno THEN                    071
            RETURN(TRUE);                                                   072
         &fz _ fz.fznext;                                                   073
         END;                                                               074
      RETURN(FALSE);                                                        075
      END.                                                                  076

%.....file name and lock message...%                                        077
   (mesfre)PROCEDURE(fl, ptr);                                              078
      REF fl;                                                               079
      dismes(2, fl.flastr);                                                 080
      IF fl.fllock THEN lockmes(ptr.stfile);                                081
      RETURN;                                                               082
      END.                                                                  083

%.....generate display.....%                                                084
   (recred)                                                                 085
      PROCEDURE;                                                            086
      IF cdtype = dspno THEN RETURN;%no display necessary%                  087
      IF NOT namereset THEN dn($"");  %clear name area%                     088
      IF cdtype = dspallf THEN %recreate all display areas%                 089
         <DSPGEN, alldsp>()                                                 090
      ELSE <DSPGEN, seldsp>();%selectively recreate display areas%          091
      RETURN;                                                               092
      END.                                                                  093
   (alldsp) PROCEDURE;  %recreate display for all display areas%            094
      %Issues Core-NLS calls to regenerate the display image                095
      for all of the currently defined text display areas.%                 096
      %-----------------%                                                   097
      LOCAL da, end, y, width;                                              098
      LOCAL STRING dtmstg[20];                                              099
      REF  da;                                                             0100
      IF dacnt NOT IN [1,damax] THEN err($"Fatal display error in          0101
```

```
   ALLDSP");                                                          0102
   IF NOT litreset THEN rstlit();                                     0103
   clearda(0);  clrall(0, TRUE);                                     02553
   end _ (&da _ $dpyarea) + dacnt*dal;                                0104
   DO IF da.daexis AND NOT da.daseq AND NOT da.dasuppress AND NOT
   da.daauxiliary THEN                                                0105
      BEGIN                                                           0106
      IF nldevice = devlproc AND (lptype .A 4M) = deltadata AND
      da.daleft NOT= taleft THEN                                     02547
         BEGIN                                                       02550
         width _ da.daright-da.daleft-1;                             02551
         FOR y _ da.datop UP da.davinc UNTIL > da.damrow DO          02548
            cline(da.daleft, y, width);                              02549
         END;                                                        02552
      da.daccnt _ 1;                                                 0107
      dafrmt(&da, 0);                                                0108
      END                                                            0109
   UNTIL (&da _ &da + dal) = end;                                    0110
   % Reset global display recreation parameters for line processor
   rstlit. %                                                         0111
   IF nldevice = devlproc THEN dpset(dspjpf, endfil, endfil, endfil);
                                                                     0112
   RETURN;                                                           0113
   END.                                                              0114
                                                                     0115
(seldsp) PROCEDURE;  %selective display recreate control%           0116
   %Issues Core-NLS calls to update or reformat the display image for
   each currently defined text display area in which a file that was
   modified is being displayed.  Reformatting is usually needed only
   if current viewspec parameters or the last viewspec change make
   selective updating impossible.%                                   0117
   %-----------------%                                              0118
   LOCAL                                                             0119
      frmt,          % call dafrmt flag %                          0120
      da,            %temp for walking thru da's%                  0122
      f1, f2,        %file numbers of files to be formated%       02269
      end;           %last word address in list of da's%           0125
   LOCAL STRING dtmstg[20];                                          0126
   REF da;                                                           0127
                                                                     0128
   %initialization%                                                  0129
      IF dacnt NOT IN [1,damax] THEN                                 0131
         err($"DACNT out of range; detected in SELDSP");            02259
      IF NOT litreset THEN rstlit();                                 0132
      %replace all stid's passed only as file indicators, not to be
      used in reformatting%                                          0140
         f1 _ IF cdstd1 = endfil THEN endfil ELSE cdstd1.stfile;
                                                                    02265
         f2 _ IF cdstd2 = endfil THEN endfil ELSE cdstd2.stfile;
                                                                    02266
         CASE cdtype OF                                              0141
            = dspjpf, = dspyes, =  dspstrc, = dspallf :             02262
               cdstd1 _ cdstd2 _ endfil;                             0142
            ENDCASE;                                                02264
      end _ (&da _ $dpyarea) + dacnt * dal;                          0143
   % search through da's %                                           0144
```

```
      DO                                                                0145
         BEGIN %locate candidate for update or reformat%                0146
         IF howformat(&da, f1, f2 : frmt ) THEN %reformat this one%
                                                                        0147
            BEGIN                                                       0156
            IF frmt THEN %redo whole display%                           0157
               dafrmt(&da, 0)                                           0158
            ELSE %selectively update display%                           0159
               daupdate (&da);                                          0160
            (da.dapvs, da.dapvs2) _ (da.davspec, da.davspc2);           0163
            END                                                         0164
         ELSE    %update window if there are suppressed statements%
                                                                        02743
            IF da.dalssup THEN daupdate(&da);                           02744
         END                                                            0165
      UNTIL (&da _ &da + dal) = end;                                    0166
   % remove bug marks from jump windows needing no recreate%           0167
      IF cdtype = dspjpf THEN bmoff();                                  0168
   % Reset global display recreation parameters%                       0169
      dpset(dspno, endfil, endfil, endfil);                             02270
   RETURN;                                                              0171
   END.                                                                 0172
                                                                        0173
(howformat) PROCEDURE (da, f1, f2);  %test if reformatting should
occur for this display area and ifso, should full reformatting occur
for this display area%                                                  02640
   LOCAL mask; %used to mask out certain viewspecs%                     02641
   REF da;                                                              02642
   IF da.daexis AND NOT da.daseq AND NOT da.daauxiliary AND NOT
   da.dasuppress AND
   (da.dacsp.stfile = f1 OR
   da.dacsp.stfile = f2 OR
   (da.davspec.vsfrzf AND frzchk(&da, f1, f2))) AND
   (cdtype NOT= dspjpf OR &da = lda())
   THEN %should be reformatted%                                         02643
      BEGIN %determine how to format it%                                02644
      mask _ -1; mask.vslev _ mask.vsrlev _ mask.vslevd _ mask.vsbrof
      _ mask.vsplxf _ 0;                                                02645
         %mask out level fields%                                        02646
      IF (((da.dapvs) .A mask) # ((da.davspec) .A mask))
         %crtical viewspecs have just changed%
      OR cdtype = dspallf
      OR cdtype = dspyes
      OR da.davspec.vscakf
      OR da.davspc2.vsmkrf
      OR da.davspec.vscapf
      OR da.davspec.vsusqf
      OR (da.davspec.vsstnf AND NOT da.davspec.vssidf AND cdtype NOT=
      dsprfmt AND cdtype # dspjpf)
      OR da.daempty
      OR (da.davspec.vsrind AND (da.davspec.vsbrof OR
      da.davspec.vsplxf) AND (cdtype NOT= dsprfmt) )
      OR (cdtype = dspjpf AND (da.dapstf # da.dacsp.stfile))
      THEN                                                              02647
         RETURN(TRUE, TRUE) %do full reformat%                          02648
      ELSE RETURN(TRUE, FALSE); %do partial reformat%                   02649
```

```
      END                                                          02650
   ELSE RETURN(FALSE, FALSE);                                      02651
   END.                                                            02652
(frzchk) PROCEDURE (da, file1, file2); %check frozen chain for file
membership%                                                        0174
   %Returns TRUE if there are anyy frozen statements from file1 or
   file2 for display area 'da'; else FALSE%                        0175
   LOCAL fz, frzflg;                                               0176
   REF da, fz;                                                     0177
   IF NOT &fz _ da.dafrzl THEN RETURN  (FALSE); %no chain, this da%
                                                                   0178
   frzflg _ FALSE; %initial value%                                 0179
   DO                                                              0180
      BEGIN                                                        0181
      IF fz.fzexis THEN                                            0182
         IF fz.fzstid.stfile = file1 OR fz.fzstid.stfile = file2 THEN
                                                                   0183
            BUMP frzflg                                            0184
         ELSE NULL                                                 0185
      ELSE err ($"illegal freeze list entry, frzchk");             0186
      END                                                          0187
   UNTIL (&fz _ fz.fznext) = 0;                                    0188
   RETURN(frzflg);                                                 0189
   END.                                                            0190
%.....build viewspec status string.....%                           0191
(curvsp) PROCEDURE( % convert viewspecs to "human" string %        0192
   vspec,    % address of viewspecs word(s) %                     0193
   astrng);  % address of string to append string  %              0194
   LOCAL vspc;                                                     0195
   REF vspec, astrng;                                              0196
   vspc _ vspec;                                                   0197
   %level%                                                         0198
      *astrng* _ *astrng*, "levels: ";                             0199
      IF vspc.vslev = 63 THEN *astrng* _ *astrng*, "ALL"           0200
      ELSE *astrng* _ *astrng*, STRING(vspc.vslev);                0201
   %truncation%                                                    0202
      *astrng* _ *astrng*, ", lines: ";                            0203
      IF vspc.vstrnc = 63 THEN *astrng* _ *astrng*, "ALL"          0204
      ELSE *astrng* _ *astrng*, STRING(vspc.vstrnc);               0205
      *astrng* _ *astrng*, ", ";                                   0206
   %branch only stuff%                                             0207
      *astrng* _ *astrng*,                                         0208
         IF vspc.vsbrof THEN 'g                                    0209
         ELSE IF vspc.vsplxf THEN 'l                               0210
            ELSE 'h;                                               0211
   %content analysis%                                              0212
      *astrng* _ *astrng*,                                         0213
         IF vspc.vscapf THEN 'i                                    0214
         ELSE IF vspc.vscakf THEN 'k                               0215
            ELSE 'j;                                               0216
   %statement numbers%                                             0217
      *astrng* _ *astrng*,                                         0218
         IF vspc.vsstnf THEN 'm ELSE 'n;                           0219
   %frozen stats%                                                  0220
      *astrng* _ *astrng*,                                         0221
         IF vspc.vsfrzf THEN 'o ELSE 'p;                           0222
```

```
    %formatter on/off%                                                  0223
       *astrng* _ *astrng*,                                             0224
          IF vspc.vsdaft THEN 'u ELSE 'v;                               0225
    %blank lines%                                                       0226
       *astrng* _ *astrng*,                                             0227
          IF vspc.vsblkf THEN 'y ELSE 'z;                               0228
    %indenting%                                                         0229
       *astrng* _ *astrng*,                                             0230
          IF vspc.vsrind THEN 'Q ELSE IF vspc.vsindf THEN 'A ELSE 'B;
                                                                       02635
    %names%                                                             0232
       *astrng* _ *astrng*,                                             0233
          IF vspc.vsnamf THEN 'C ELSE 'D;                               0234
    %Pagination%                                                        0235
       *astrng* _ *astrng*,                                             0236
          IF vspc.vspagf THEN 'E ELSE 'F;                               0237
    %stat nums left/right%                                              0238
       *astrng* _ *astrng*,                                             0239
          IF vspc.vsstnr THEN 'G ELSE 'H;                               0240
    %stat nums or SID's%                                                0241
       *astrng* _ *astrng*,                                             0242
          IF vspc.vssidf THEN 'I ELSE 'J;                               0243
    %signature%                                                         0244
       *astrng* _ *astrng*,                                             0245
          IF vspc.vsidtf THEN 'K ELSE 'L;                               0246
    %user sequence generator%                                           0247
       *astrng* _ *astrng*,                                             0248
          IF vspc.vsusqf THEN 'O ELSE 'P;                               0249
    RETURN END.
                                                                        0250
%.....freeze statement support.....%                                    0251
  (xrelsta)                                                             0252
    %This routine searches the chain of frozen statements.              0253
    If the stid passed it is in the frozen list for the display         0254
    area passed it, the routine removes it from the list,               0255
    squeezes the frozen list, and adds the deleted element to           0256
    the fozen element free list.%                                       0257
    %------------------%                                               0258
    PROCEDURE(dpa, stid);                                               0259
    LOCAL frzprev, frzelm;                                              0260
    REF dpa, frzprev, frzelm;                                           0261
    IF &frzprev _ &frzelm _ dpa.dafrzl THEN LOOP                        0262
       BEGIN                                                            0263
       IF frzelm.fzstid = stid THEN                                     0264
          BEGIN                                                         0265
          IF &frzelm = &frzprev THEN %first item in list%               0266
             dpa.dafrzl _ frzelm.fznext                                 0267
          ELSE frzprev.fznext _ frzelm.fznext;                          0268
          frzelm.fzexis _ FALSE;                                        0269
          frzelm.fznext _ fzfree := &frzelm;                            0270
          EXIT;                                                         0271
          END;                                                          0272
       &frzprev _ &frzelm;                                              0273
       IF frzelm.fznext THEN &frzelm _ frzelm.fznext                    0274
       ELSE EXIT;                                                       0275
       END;                                                             0276
```

```
   RETURN;                                                          0277
   END.                                                             0278
(xrelallsta)                                                        0279
   %Given the address of a display area, this routine will          0280
   free all of the frozen elements associated with the area.%       0281
   %------------------%                                             0282
   PROCEDURE(dpa);                                                  0283
   LOCAL frzelm;                                                    0284
   REF dpa, frzelm;                                                 0285
   IF &frzelm _ dpa.dafrzl THEN                                     0286
      BEGIN                                                         0287
      LOOP                                                          0288
         BEGIN                                                      0289
         frzelm.fzexis _ FALSE;                                     0290
         IF NOT frzelm.fznext THEN EXIT                             0291
         ELSE &frzelm _ frzelm.fznext;                              0292
         END;                                                       0293
      frzelm.fznext := fzfree := dpa.dafrzl := 0;                   0294
      END;                                                          0295
   RETURN;                                                          0296
   END.                                                             0297
                                                                    0298
(xfresta)                                                           0299
   %Given the addrss of a display are, an stid, and two             0300
   viewspec words, this routine will create a frozen element        0301
   for the stid passed it, in the display area passed.  If          0302
   the stid is already on the frozen lit for this area, the         0303
   new vspec words will replace the old ones for that element.%     0304
   %------------------%                                             0305
   PROCEDURE(dpa, stid, vspec1, vspec2);                            0306
   LOCAL frzelm;                                                    0307
   REF dpa, frzelm;                                                 0308
   IF &frzelm _ dpa.dafrzl THEN                                     0309
      BEGIN                                                         0310
      LOOP                                                          0311
         BEGIN                                                      0312
         IF frzelm.fzstid = stid THEN                               0313
            BEGIN                                                   0314
            frzelm.fzvspec _ vspec1;                                0315
            frzelm.fzvspc2 _ vspec2;                                0316
            RETURN;                                                 0317
            END;                                                    0318
         IF frzelm.fznext THEN &frzelm _ frzelm.fznext              0319
         ELSE EXIT;                                                 0320
         END;                                                       0321
      IF NOT fzfree THEN err(5);                                    0322
      &frzelm _ frzelm.fznext _ fzfree := [fzfree].fznext;          0323
      frzelm.fznext _ 0;                                            0324
      END                                                           0325
   ELSE                                                             0326
      BEGIN                                                         0327
      IF NOT fzfree THEN err(5);                                    0328
      dpa.dafrzl _ &frzelm _ fzfree := [fzfree].fznext;             0329
      frzelm.fznext _ 0;                                            0330
      END;                                                          0331
   frzelm.fzstid _ stid;                                            0332
```

```
        frzelm.fzvspec _ vspec1;                                        0333
        frzelm.fzvspc2 _ vspec2;                                        0334
        frzelm.fzexis _ TRUE;                                           0335
        RETURN;                                                         0336
        END.                                                            0337
                                                                        0338
%.....simulate device type support.....%                                0339
    (xsimdev)    %Execute simulate Device Command%                      0340
        PROCEDURE(da, devicetype, modetype);                            0341
        REF da;                                                         0342
        REF rawchr, msgda, tda;                                        01872
        LOCAL oldnlmode; %for saving ced entry nlmode setting
                          (changed by setdev)%                          0343
        oldnlmode _ nlmode;                                             0344
        IF devicetype NOT= nldevice OR modetype NOT= nlmode THEN        0345
            BEGIN                                                       0346
            continue _ TRUE;                                           01875
            IF modetype NOT= nlmode THEN                                0347
                BEGIN                                                   0348
                IF nlmode = fulldisplay THEN shutdis();                 0349
                <INTNLS, setdev> (devicetype);                          0350
                initch(devicetype);                                     0351
                initbtbl();                                             0352
                IF oldnlmode = typewriter THEN % TNLS --> DNLS %        0353
                    BEGIN                                               0354
                    IF savedda THEN  %came from DNLS before -- restore old
                    da%                                                 0355
                        BEGIN                                           0356
                        &tda _ savedda := 0;                            0357
                        tda.dasuppress _ FALSE;                         0358
                        tda.dacsp _ da.dacsp;                           0359
                        tda.daccnt _ 1;                                 0360
                        tda.davspec _ da.davspec;                       0361
                        tda.davspc2 _ da.davspc2;                       0362
                        tda.dalink _ da.dalink := tda.dalink;           0363
                        tda.dafrzl _ da.dafrzl := 0;                    0364
                        tda.dapstf _ da.dapstf;                         0365
                        tda.dacacode _ da.dacacode;                     0366
                        tda.dausqcod _ da.dausqcod;                     0367
                        tda.daukeycod _ da.daukeycod;                   0368
                        delda(&da := 0);                                0369
                        END                                             0370
                    ELSE  %first time%                                  0371
                        BEGIN                                           0372
                        continue _ FALSE;                              01876
                        savetda _ &da;                                  0373
                        &tda _ newda();                                 0374
                        <INTNLS, intdafl> (&tda);                       0375
                        tda.dacsp _ da.dacsp;                           0376
                        tda.daccnt _ 1;                                 0377
                        tda.davspec _ da.davspec;                       0378
                        tda.davspc2 _ da.davspc2;                       0379
                        tda.dalink _ da.dalink := tda.dalink;           0380
                        tda.dafrzl _ da.dafrzl := 0;                    0381
                        tda.dapstf _ da.dapstf;                         0382
                        tda.dacacode _ da.dacacode;                     0383
```

```
            tda.dausqcod _ da.dausqcod;                                  0384
            tda.daukeycod _ da.daukeycod;                                0385
            da.dasuppress _ TRUE;                                        0386
            inittimer();                                                 0387
            END;                                                         0388
         END                                                             0389
      ELSE % DNLS --> TNLS %                                             0390
         BEGIN                                                           0391
         IF savetda THEN  %came from TNLS before -- restore old
         da%                                                             0392
            BEGIN                                                        0393
            &tda _ savetda := 0;                                         0394
            tda.dasuppress _ FALSE;                                      0395
            tda.dacsp _ da.dacsp;                                        0396
            tda.daccnt _ 1;                                              0397
            tda.davspec _ da.davspec;                                    0398
            tda.davspc2 _ da.davspc2;                                    0399
            tda.dalink _ da.dalink := tda.dalink;                        0400
            tda.dafrzl _ da.dafrzl := 0;                                 0401
            tda.dapstf _ da.dapstf;                                      0402
            tda.dacacode _ da.dacacode;                                  0403
            tda.dausqcod _ da.dausqcod;                                  0404
            tda.daukeycod _ da.daukeycod;                                0405
            delda(&da := 0);                                             0406
            END                                                          0407
         ELSE  %first time%                                              0408
            BEGIN                                                        0409
            savedda _ &da;                                               0410
            &tda _ newda();                                              0411
            <INTNLS, intdafl> (&tda);                                    0412
            tda.dacsp _ da.dacsp;                                        0413
            tda.daccnt _ 1;                                              0414
            tda.davspec _ da.davspec;                                    0415
            tda.davspc2 _ da.davspc2;                                    0416
            tda.dalink _ da.dalink := tda.dalink;                        0417
            tda.dafrzl _ da.dafrzl := 0;                                 0418
            tda.dapstf _ da.dapstf;                                      0419
            tda.dacacode _ da.dacacode;                                  0420
            tda.dausqcod _ da.dausqcod;                                  0421
            tda.daukeycod _ da.daukeycod;                                0422
            da.dasuppress _ TRUE;                                        0423
            END;                                                         0424
         vspsav _ vspsav[1] _ 0; %viewspecs will be refreshed upon
         reentry%                                                        0425
         END;                                                            0426
      END;                                                               0427
   initdis();                                                            0428
   continue _ FALSE;                                                    01874
   IF nlmode = fulldisplay THEN dsubsys( $ssysname );                   01878
   END;                                                                  0429
RETURN;                                                                 01871
END.

                                                                        0433
%.....record session support.....%                                      0434
(ctlquit) PROCEDURE;  %shut off control stuff%                          0435
   %close control file (if one is being used), reinitialize control
```

```
   jfn's, and reset subsystem name%                                          0436
   %---------------%                                                        0437
   LOCAL jfn;                                                               0438
   &rawchr _ IF nldevice # devlproc THEN $getchar ELSE                      0439
      IF altinp.L THEN $lpaltgetchar ELSE $lpgetchar;                      02351
      %set lowest level input routine to be simple get-a-character
      routine%                                                              0440
   IF inpwatchjfn THEN jfn _ inpwatchjfn  %recording session%               0441
   ELSE                                                                     0442
      IF inpjfn THEN jfn _ inpjfn %playback session%                        0443
      ELSE RETURN;                                                          0444
   IF NOT sysclose(jfn, $lit) THEN dismes(2, $lit);                         0445
   inpwatchjfn _ inpjfn _ 0;                                                0446
   r1 _ nlssbn _ getsbn($"DNLS");                                           0447
      %if control environment ever run from tty or net, should be
      smart enough to figure out real name%                                 0448
   !JSYS setnm;                                                             0449
   RETURN;                                                                  0450
   END.

                                                                            0451
(xrecplasup) %record or playback from file%                                 0452
   PROCEDURE(record, f1, f2, rhostn);                                       0453
   LOCAL jfn, gtjflg, opnflg;                                               0454
   LOCAL STRING fname[80];                                                  0455
   REF f1, f2;                                                              0456
   CASE rhostn OF                                                          01751
      = lhostn: NULL;                                                      01752
      ENDCASE err($"Remote File Manipulations Not Implemented Yet");
                                                                           01753
   IF record THEN %recording input for later playback%                      0457
      BEGIN                                                                 0458
      opnflg _ write;                                                       0459
      gtjflg _ gtjoof;                                                      0460
      END                                                                   0461
   ELSE                                                                     0462
      BEGIN                                                                 0463
      opnflg _ read;                                                        0464
      gtjflg _ gtjoif;                                                      0465
      END;                                                                  0466
   *fname* _ f1 f2;                                                         0467
   IF NOT jfn _ lgetjfn(0, $fname, $ctlext, gtjflg, $lit) THEN              0468
      err($lit);                                                            0469
   IF NOT sysopen(jfn, opnflg, chrtyp, $lit) THEN                           0470
      BEGIN                                                                 0471
      reljfn(jfn);                                                          0472
      err($lit);                                                            0473
      END;                                                                  0474
   IF record THEN                                                           0475
      BEGIN                                                                 0476
      inpwatchjfn _ jfn;                                                    0477
      %for bin/bout zero byte anomoly%                                      0478
         r1 _ jfn;                                                          0479
         r2 _ 1;                                                            0480
         !JSYS bout;                                                        0481
      END                                                                   0482
   ELSE                                                                     0483
```

```
        BEGIN                                                          0484
        inpjfn _ jfn;                                                  0485
        r1 _ nlssbn _ <AUXCOD, getsbn>($"DNLCTL");                     0486
        %tor bin/bout zero byte anomoly%                               0487
           r1 _ jfn;                                                   0488
           !JSYS bin;                                                  0489
        END;                                                           0490
     &rawchr _ $cntrlgetchar;                                          0491
        %change lowest level input routine to be one which knows about
        control stuff%                                                 0492
     RETURN;                                                           0493
     END.
                                                                       0494
%.....auxilliary input stuff....%                                      0495
   (auxstartup) %***% PROCEDURE( %setup to do input from auxilary
   source%                                                            02559
      ptr1, %pointer to TP for start of auxilary text%                02560
      ptr2); %pointer to TP for end of auxilary text%                 02561
      LOCAL fl, stdblk;                                               02562
      REF rawchr; %****take out when through debugging%               02819
      REF ptr1, ptr2, fl;                                             02563
       % change dispatch for "rawchr" to auxilary routine %           02564
        auxsav _ &rawchr;  %save current character dispatch%          02565
        &rawchr _ $auxchr; %and subsitute mine(1st time guy)%         02566
      % release old statement if left around by mistake%             02820
        IF auxwrk AND NOT auxwrk.stastr THEN                          02821
           BEGIN                                                      02822
           stdblk _ lodprop(auxwrk, txttyp);                          02823
           frzblk(stdblk,-1);                                         02824
           END;                                                       02825
      % setup work area and save text pointers %                     02567
        auxwrk _ auxtp1 _ ptr1;                                       02568
        auxwrk[1] _ auxtp1[1] _ ptr1[1];                              02569
        auxtp2 _ ptr2;                                                02570
        auxtp2[1] _ ptr2[1];                                          02571
      % nail down file if not a-string %                             02572
        IF NOT auxwrk.stastr THEN                                     02573
           BEGIN                                                      02574
           &fl _ flntadr(auxwrk.stfile);                              02575
           fl.flnoclos _ TRUE;                                        02576
           %freeze statement text%                                    02830
              stdblk _ lodprop(auxwrk, txttyp);                       02828
              frzblk(stdblk, 1);                                      02829
              %an stid in auxwrk is a flag that the sdb is frozen in
              NLS%                                                    02831
           %pass over statement name %                                02827
              auxwrk[1] _ fchtxt(auxwrk);                             02826
           END;                                                       02577
      % setup work area %                                            02578
        fechc1(forward, $auxwrk);                                     02580
      % jam recognition mode to be "demand" %                        02581
        auxmod _ recogmode; %1st save current one %                   02582
        auxmd2 _ recog2mode;                                          02583
        recogmode _ mdemand;                                          02584
        recog2mode _ mdemand;                                         02585
        auxinput _ TRUE;                                              02692
```

```
   % force to recognize upper/lower case %                          02809
      CASE nldevice OF                                              02810
         = dev33, = dev35, =offline:  initch(nettty);               02811
         ENDCASE;                                                   02812
   RETURN;                                                          02586
   END.                                                             02587
(auxchr) %***% PROCEDURE; %get characters for executable text%
                                                                    02588
   LOCAL char, stdblk;                                              02589
   REF rawchr; %****take out when through debugging%                02842
   LOOP                                                             02590
      IF (NOT inptrf) AND (auxwrk # auxtp2 OR POS auxwrk < auxtp2)
      THEN %characters left to get%                                 02591
         BEGIN                                                      02592
         %will this make process commands run faster ?%             02593
            IF tenex < 13200 THEN                                   02630
               BEGIN                                                02631
                r1 _ 0;                                             02632
               !JSYS simch;                                         02633
               END;                                                 02634
         % The JSYS makes the fork appear to the scheduler as though
         it were just unblocked for character input%                02596
         ON SIGNAL ELSE auxinterminate();                           02597
         CASE (char _ READC($auxwrk)) OF     %get the next char%
                                                                    02598
            = ENDCHR:                                               02599
               IF NOT basestateflag THEN                            02600
                  BEGIN                                             02601
                  % continue with normal input if end of statement
                  and in parse or execution%                        02602
                  auxinterminate(); %go terminate auxilary input%
                                                                    02603
                  RETURN(rawchr());                                 02604
                  END                                               02605
               ELSE                                                 02606
                  BEGIN                                             02607
                  % release old statement%                          02832
                     IF auxwrk AND NOT auxwrk.stastr THEN           02833
                        BEGIN                                       02834
                        stdblk _ lodprop(auxwrk, txttyp);           02835
                        frzblk(stdblk,-1);                          02836
                        END;                                        02837
                  auxwrk _ getnxt(auxwrk);                          02608
                  %freeze statement text%                           02838
                     stdblk _ lodprop(auxwrk, txttyp);              02839
                     frzblk(stdblk, 1);                             02840
                     %an stid in auxwrk is a flag that the sdb is
                     frozen in NLS%                                 02841
                  auxwrk[1] _ fchtxt(auxwrk);                       02609
                  fechc1(forward, $auxwrk);                         02610
                  REPEAT LOOP;                                      02611
                  END                                               02612
            ENDCASE                                                 02613
               BEGIN                                                02614
               IF echofg THEN typech(char);                         02615
               RETURN(char);                                        02616
```

```
                END;                                                    02617
            END                                                         02618
         ELSE                                                           02619
            BEGIN                                                       02620
            auxinterminate(); %go terminate auxilary input%             02621
            IF inptrf THEN                                              02622
               BEGIN                                                    02623
               dismes(2, $"User Terminated Input");                     02624
               RETURN(CD)                                               02625
               END                                                      02626
            ELSE RETURN(rawchr());                                      02627
            END;                                                        02628
      END.                                                              02629
   (auxinterminate) PROCEDURE; %terminate auxilary input%                0558
      LOCAL fl, stdblk;                                                 0559
      REF rawchr; %****take out when through debugging%                 02851
      REF fl;                                                           0560
      % reset recognition mode and char input routine %                  0561
         &rawchr _ auxsav;       %reset where to get characters%        0562
         recogmode _ auxmod;     %reset recognition mode%               0563
         recog2mode _ auxmd2;                                           0564
         auxinput _ FALSE;                                              02693
      % free up file if not a-string %                                   0565
         IF NOT auxwrk.stastr THEN                                      0566
            BEGIN                                                       0567
            &fl _ flntadr(auxwrk.stfile);                               0568
            fl.flnoclos _ FALSE;                                        0569
            % release old statement%                                   02843
               stdblk _ lodprop(auxwrk, txttyp);                        02846
               frzblk(stdblk,-1);                                       02847
            % flag it as released %                                    02849
               auxwrk _ 0;                                              02850
            END;                                                        0570
      % reset upper/lower case translation %                           02813
         CASE nldevice OF                                               02814
            =dev33, =dev35, =offline:  initch(nldevice);                02815
            ENDCASE;                                                    02816
      RETURN;                                                           0571
      END.                                                              0572
%.....screen spliting Support Routines.....%                            0573
   (vsplit) %split window vertically%
   PROCEDURE (left, cords1, cords2);                                    0574
      %split the display area (left) vertically, at the column
      determined by cords1 and move the old display image to the right
      or left depending on the cursor position for the final command
      accept(passed as cords2).%                                        0575
      %---------------%                                                 0576
      LOCAL                                                             0577
         right,    %A(da entry for right (new) da)%                     0578
         lines,    %number of lines in each da%                         0579
         rcolumns, %number of columns in right da%                      0580
         lcolumns, %number of columns in left da%                       0581
         entry,    %utility cell%                                       0582
         count,    %utility cell%                                       0583
         lbnd,     %left da boundry + 20 chars%                         0584
         rbnd;     %right da boundry - 20 chars%                        0585
```

```
      %min da is two lines by 20 columns%                              0586
REF left, right;                                                       0587
%Note:  this version of vsplit allows as many columns as will fit
in window, i.e. damcol = daright%
                                                                       0588
DMOFF();                                                               0589
%check for valid split%                                                0590
   lbnd _ left.daleft + (20*left.dahinc);                              0591
   rbnd _ left.daright - (20*left.dahinc);                             0592
   IF cords1.xcord < lbnd THEN                                         0593
      cords1.xcord _ lbnd;                                             0594
   IF cords1.xcord > rbnd THEN                                         0595
      cords1.xcord _ rbnd;                                             0596
   IF cords1.xcord NOT IN [lbnd,rbnd] THEN                             0597
      RETURN(FALSE);                                                   0598
&right _ newda();                                                      0599
   %get address of new da entry%                                       0600
%copy old da entry to new one%                                         0601
   copyda(&left, &right);                                              0602
%update da boundries%                                                  0603
   right.daleft _ left.daright _ cords1.xcord;                         0604
%update max columns -- if wrapping, use the max (of window width
or display right margin); if not wrapping, use the min%          02685
   right.damcol _ (right.daright - right.daleft)/right.dahinc;
                                                                 02686
   IF (right.damcol > udpcolmax AND udpcolmax>0) OR (udpwrapcol>0
   AND udpwrapcol < right.damcol) THEN                           02687
      right.damcol _ udpcolmax*right.dahinc;                     02688
   left.damcol _ (left.daright - left.daleft)/left.dahinc;       02689
   IF (left.damcol > udpcolmax AND udpcolmax>0) OR (udpwrapcol>0
   AND udpwrapcol < left.damcol) THEN                            02690
      left.damcol _ udpcolmax*left.dahinc;                       02691
%update display buffer pointers%                                       0608
   lines _ (left.dabottom-left.datop)/left.davinc;                     0609
%get new display list entries%                                         0614
   IF rtfree + MIN(60, lines*3) > rtlast THEN                          0615
      err($"Not enough line segments, hsplit");                        0616
   rtfree _ rtfree + MIN(lines * 3, 60);                               0617
   %reallocate old LSRT space%                                         0618
   freelsrt ($dspblk, &left);                                          0619
   IF NOT maklsrt($dspblk, lines + 2, &left) THEN                      0620
      err($"NLS out of space for LSRT");                               01787
   %allocate space for new LSRT%                                       0621
   IF NOT maklsrt($dspblk, lines + 2, &right) THEN                     0622
      err($"NLS out of space for LSRT");                               01788
IF cords2.xcord > cords1.xcord THEN %right gets old display%           0623
   BEGIN  %zap left side%                                              0624
   left.dacsp _ endfil;                                                0625
      %zap csp, frzlist, and link ring for this da%                    0626
   left.dafrzl _ 0;                                                    0627
   left.daempty _ TRUE;                                                0628
      %empty window--nothing being displayed in it%                    0629
   left.dalink _ right.dalink := left.dalink;                          0630
   END                                                                 0631
ELSE %left gets old display%                                           0632
   BEGIN  %zap right side%                                             0633
```

```
      right.dacsp _ endfil;                                          0634
         %zap csp, frzlist, and link ring for this da%               0635
      right.dafrzl _ 0;                                              0636
      right.daempty _ TRUE;                                          0637
         %empty window--nothing being displayed in it%               0638
      END;                                                           0639
   left.darneighbor _ (&right - $dpyarea)/dal + 1;                   01789
   right.dalneighbor _ (&left - $dpyarea)/dal + 1;                   01832
   %deallocate old da and get two new ones%                          01761
      dealocda(&left);                                               01762
      right.dahandle _ right.dalhandle _ 0;                          01763
      <NLS, DSPGEN, alocda>(&left);                                  01764
      <NLS, DSPGEN, alocda>(&right);                                 01765
   RETURN(TRUE);                                                     0658
   END.                                                              0659
                                                                     0660
(hsplit) %split window horizontally%
PROCEDURE (top, cords1, cords2);                                     0661
   %split the display area (top) horizontally, at the row determined
   by cords1 and move the old display to the top or bottom da
   depending on the cursor position for the final command accept
   (passed as cords2).%                                              0662
   %---------------%                                                0663
   LOCAL                                                             0664
      bottom,   %A(da entry for bottom (new) da)%                    0665
      columns,  %number of columns in each da%                       0666
      tlines,   %number of lines in top da%                          0667
      blines,   %number of lines in bottom da%                       0668
      entry,    %utility cell%                                       0669
      count,    %utility cell%                                       0670
      tbnd,     %top da boundry + 1 line%                            0671
      bbnd;     %bottom da boundry - 1 line%                         0672
         %min da is two lines by 20 columns%                         0673
   REF top, bottom;                                                  0674
   bmoff();                                                          0675
   %Note:  This version of hsplit allows as many rows as will fit on
   screen, i.e. damrow value = dabottom - datop  value%
                                                                     0676
   %check for valid split%                                           0677
      tbnd _ top.datop + (2*top.davinc);                             0678
      bbnd _ top.dabottom - (2*top.davinc);                          0679
      IF cords1.ycord < tbnd THEN                                    0680
         cords1.ycord _ tbnd;                                        0681
      IF cords1.ycord > bbnd THEN                                    0682
         cords1.ycord _ bbnd;                                        0683
      IF cords1.ycord NOT IN [tbnd,bbnd] THEN                        0684
         RETURN(FALSE);                                              0685
   &bottom _ newda();                                                0686
      %get address of new da entry%                                  0687
   %copy old da entry to new one%                                    0688
      copyda(&top, &bottom);                                         0689
   %update da boundries%                                             0690
      bottom.datop _ top.dabottom _ cords1.ycord;                    0691
   %update max lines%                                                0693
      blines _ (bottom.dabottom - bottom.datop)/bottom.davinc;       0694
         bottom.damrow _  bottom.dabottom - bottom.datop; %use all of
```

```
      available lines on screen%                                    0699
   tlines _ (top.dabottom - top.datop)/top.davinc;                  0700
      top.damrow _ top.dabottom - top.datop - top.davinc; %one
      line less than top of bottom window%                          0692
%get rid of extra space on top LSRT%                                0705
   freelsrt($dspblk, &top);                                         0706
   IF NOT maklsrt($dspblk, tlines + 2, &top) THEN                   0707
      err($"NLS out of space for LSRT");                            01790
%get new display list entries for da called "bottom" %              0708
   IF NOT maklsrt($dspblk, blines + 2, &bottom) THEN                0709
      err($"NLS out of space for LSRT");                            01791
IF cords2.ycord > cords1.ycord THEN %bottom gets old display%       0710
   BEGIN  %zap top side%                                            0711
   top.dacsp _ endfil;                                              0712
      %zap csp, frzlist, and link ring for this da%                 0713
   top.dafrzl _ 0;                                                  0714
   top.daempty _ TRUE;                                              0715
      %empty window--nothing being displayed in it%                 0716
   top.dalink _ bottom.dalink := top.dalink;                        0717
   END                                                              0718
ELSE %top gets old display%                                         0719
   BEGIN  %zap bottom side%                                         0720
   bottom.dacsp _ endfil;                                           0721
      %zap csp, frzlist, and link ring for this da%                 0722
   bottom.dafrzl _ 0;                                               0723
   bottom.daempty _ TRUE;                                           0724
      %empty window--nothing being displayed in it%                 0725
   END;                                                             0726
top.dabneighbor _ (&bottom - $dpyarea)/dal + 1;                     01792
bottom.datneighbor _ (&top - $dpyarea)/dal + 1;                     01833
%deallocate old da and get two new ones%                            01766
   dealocda(&top);                                                  01767
   bottom.dahandle _ bottom.dalhandle _ 0;                          01768
   <NLS, DSPGEN, alocda>(&top);                                     01769
   <NLS, DSPGEN, alocda>(&bottom);                                  01770
RETURN(TRUE);                                                       0745
END.                                                                0746
                                                                    0747
(boundry) %find nearest window edge%
PROCEDURE (dacords, margin);                                        02694
   %Returns dano and coordinates for the point on the boundry       02695
   nearest the passed coordinates and the type of boundry (tbound,
                                                                    02696
   bbound, lbound, rbound).  If margin is TRUE, then margins are
   acceptable boundaries; otherwise the nearest internal boundary is
   found (if none exist, it calls err).%                            02697
   %------------------%                                            02698
   LOCAL                                                            02699
      da, top, bottom, left, right, dano;                           02700
   REF da;                                                          02701
   IF dacnt NOT IN [1,damax] THEN                                   02702
      err($"Illegal dacnt, boundry");                               02703
   IF NOT &da _ dsparea(dano _ findda(dacords : dacords.xcord,
   dacords.ycord)) THEN                                             02704
      err($"findda failed, boundry");                               02705
   top _ dacords.ycord - da.datop;                                  02706
```

```
   bottom _ da.dabottom - dacords.ycord;                                02707
   lett _ dacords.xcord - da.daleft;                                    02708
   right _ da.daright - dacords.xcord;                                  02709
   CASE MIN(left, right, top, bottom) OF                                02710
      = 377777B:  err($"No internal edge found.");                      02711
      = left:                                                           02712
         BEGIN                                                          02713
         dacords.xcord _ da.daleft;                                     02714
         IF margin OR da.daleft # taleft THEN RETURN(dano, dacords,
         lbound);    %type of boundry is lbound%                        02715
         left _ 377777B;     %a large number%                           02716
         REPEAT CASE;        %to check the next nearest boundary%
                                                                        02717
         END;                                                           02718
      = right:                                                          02719
         BEGIN                                                          02720
         dacords.xcord _ da.daright;                                    02721
         IF margin OR da.daright # taright THEN RETURN(dano, dacords,
         rbound);    %type of boundry is rbound%                        02722
         right _ 377777B;    %a large number%                           02723
         REPEAT CASE;        %to check the next nearest boundary%
                                                                        02724
         END;                                                           02725
      = top:                                                            02726
         BEGIN                                                          02727
         dacords.ycord _ da.datop;                                      02728
         IF margin OR da.datop # tatop THEN RETURN(dano, dacords,
         tbound);    %type of boundry is tbound%                        02729
         top _ 377777B;      %a large number%                           02730
         REPEAT CASE;        %to check the next nearest boundary%
                                                                        02731
         END;                                                           02732
      = bottom:                                                         02733
         BEGIN                                                          02734
         dacords.ycord _ da.dabottom;                                   02735
         IF margin OR da.dabottom # tabottom THEN RETURN(dano,
         dacords, bbound);   %type of boundry is bbound%                02736
         bottom _ 377777B;   %a large number%                           02737
         REPEAT CASE;        %to check the next nearest boundary%
                                                                        02738
         END;                                                           02739
      ENDCASE;                                                          02740
   END.                                                                 02741
                                                                        02742

(movbndry) %move window boundary%
PROCEDURE (da1, ocords, ncords, type);                                   0797
   %The cordinates 'ocords' points to a boundry point (see boundry)
   and 'ncords' contains the cordinates of the user's second
   selection.  Boundries can be moved at most as far as the nearest
   neighbor in the direction of motion.  If the selected boundry is
   between two da's whose boundries are the same in the orthogonal
   direction, the boundry is adjusted only between those two da's.
   Otherwise all da's who share that boundry will be adjusted.
   Boundries at the edge of the screen can not be moved.  Buffer
   space will be reapportioned for all da's after a boundry
   adjustment.%                                                         0799
```

```
   %------------------%                                                0812
   LOCAL                                                              0813
      da2, dacord2, dano2;                                            0814
   REF da1, da2;                                                      0815
   bmoff();                                                          01793
   %find neighbor on other side of selected boundry%                  0817
      dacord2 _ ocords;                                               0818
      CASE type OF                                                    0819
         = tbound: %top boundry selected%                             0820
            dacord2.ycord _ da1.datop - 1;                            0821
         = bbound: %bottom boundry selected%                          0822
            dacord2.ycord _ da1.dabottom + 1;                         0823
         = lbound: %left boundry selected%                            0824
            dacord2.xcord _ da1.daleft - 1;                           0825
         = rbound: %right boundry selected%                           0826
            dacord2.xcord _ da1.daright + 1;                          0827
         ENDCASE err($"Illegal boundary type, moveboundary");         0828
      IF NOT (&da2 _ dsparea(dano2 _ findda(dacord2))) OR
      &da1 = &da2 THEN                                                0829
         err($"Invalid move request");                                0830
   CASE type OF                                                       0831
      = tbound: %top boundry of da1, bottom of da2%                   0832
         IF da1.daleft = da2.daleft AND                               0833
         da1.daright = da2.daright THEN                               0834
            BEGIN %just move this boundry%                            0835
            IF ncords.ycord < da2.datop OR
            (ncords.ycord-da2.datop)/da2.davinc < 2 THEN              0836
               ncords.ycord _ da2.datop + (2*da2.davinc)              0837
            ELSE                                                      0838
               IF ncords.ycord > da1.dabottom OR
               (da1.dabottom-ncords.ycord)/da1.davinc < 2 THEN        0839
                  ncords.ycord _ da1.dabottom - (2*da1.davinc);       0840
            da1.datop _ da2.dabottom _ (ncords.ycord/tavinc)*tavinc;
                                                                      0841
            END                                                       0842
         ELSE  %adjust all da's which share this boundry%             0843
            fixbnd(FALSE, da1.datop, ncords.ycord, FALSE);            0844
      = bbound: %bottom boundry of da1, top of da2%                   0845
         IF da1.daleft = da2.daleft AND                               0846
         da1.daright = da2.daright THEN                               0847
            BEGIN %just move this boundry%                            0848
            IF ncords.ycord < da1.datop OR
            (ncords.ycord-da1.datop)/da1.davinc < 2 THEN              0849
               ncords.ycord _ da1.datop + (2*da1.davinc)              0850
            ELSE                                                      0851
               IF ncords.ycord > da2.dabottom OR
               (da2.dabottom-ncords.ycord)/da2.davinc < 2 THEN        0852
                  ncords.ycord _ da2.dabottom - (2*da2.davinc);       0853
            da1.dabottom _ da2.datop _ (ncords.ycord/tavinc)*tavinc;
                                                                      0854
            END                                                       0855
         ELSE  %adjust all da's which share this boundry%             0856
            fixbnd(FALSE, da1.dabottom, ncords.ycord, FALSE);         0857
      = lbound: %left boundry of da1, right of da2%                   0858
         IF da1.datop = da2.datop AND                                 0859
         da1.dabottom = da2.dabottom THEN                             0860
```

```
            BEGIN %just move this boundry%                                  0861
            IF ncords.xcord > da1.daright OR
            (da1.daright-ncords.xcord)/da1.dahinc < 20 THEN                 0862
               ncords.xcord _ da1.daright - (20*da1.dahinc)                 0863
            ELSE                                                            0864
               IF ncords.xcord < da2.daleft OR
               (ncords.xcord-da2.daleft)/da2.dahinc < 20 THEN               0865
                  ncords.xcord _ da2.daleft + (20*da2.dahinc);              0866
            da1.daleft _ da2.daright _ (ncords.xcord/tahinc)*tahinc;
                                                                            0867
            END                                                             0868
         ELSE  %adjust all da's which share this boundry%                   0869
            fixbnd(TRUE, da1.daleft, ncords.xcord, FALSE);                  0870
      = rbound: %right boundry of da1, left of da2%                         0871
         IF da1.datop = da2.datop AND                                       0872
         da1.dabottom = da2.dabottom THEN                                   0873
            BEGIN %just move this boundry%                                  0874
            IF ncords.xcord < da1.daleft OR
            (ncords.xcord-da1.daleft)/da1.dahinc < 20 THEN                  0875
               ncords.xcord _ da1.daleft + (20*da1.dahinc)                  0876
            ELSE                                                            0877
               IF ncords.xcord > da2.daright OR
               (da2.daright-ncords.xcord)/da2.dahinc < 20 THEN              0878
                  ncords.xcord _ da2.daright - (20*da2.dahinc);             0879
            da1.daright _ da2.daleft _ (ncords.xcord/tahinc)*tahinc;
                                                                            0880
            END                                                             0881
         ELSE  %adjust all da's which share this boundry%                   0882
            fixbnd(TRUE, da1.daright, ncords.xcord, FALSE);                 0883
      ENDCASE;                                                              0884
   fixbuf();                                                                0885
      %also deletes windows that are too small%                             0886
   RETURN;                                                                  0887
   END.                                                                     0888
                                                                            0889
(fixbnd) %fix boundaries%
PROCEDURE (type, value, newvalue, allowdelete);                             0890
   % if type  is true then vertical boundry, otherwise horizontal. If
   newvalue is farther from value than nearest neighbor, then set
   newvalue to minus minimum size (20 colums or 2 lines).  Search
   display area record for vertical/horizontal boundries = value and
   replace them by newvalue%                                                0891
   %---------------%                                                        0897
   LOCAL da, end;                                                           0898
   REF da;                                                                  0899
   IF dacnt NOT IN [2,damax] THEN                                           0900
      err($"cannot move or delete margin edges");                           0901
   end _ (&da _ $dpyarea) + dacnt*dal;                                      0902
   DO IF da.daexis THEN                                                     0903
      IF type THEN %vertical boundry%                                       0904
         BEGIN                                                             02389
         IF da.daleft = value THEN                                          0905
            BEGIN                                                          02324
            IF allowdelete THEN                                            02353
               BEGIN                                                       02367
               IF newvalue > da.daright THEN                                0906
```

```
                newvalue _ da.daright                                    0907
             END                                                         02368
          ELSE                                                           02354
             BEGIN                                                       02370
             IF newvalue > da.daright - (20*da.dahinc) THEN              02355
                newvalue _ da.daright - (20*da.dahinc)                   02356
             END                                                         02369
          END                                                            0908
       ELSE                                                              0909
          BEGIN                                                          02365
          IF da.daright = value THEN                                     0910
             BEGIN                                                       02373
             IF allowdelete THEN                                         02357
                BEGIN                                                    02372
                IF newvalue < da.daleft THEN                             0911
                   newvalue _ da.daleft                                  0912
                END                                                      02371
             ELSE                                                        02358
                BEGIN                                                    02376
                IF newvalue < da.daleft + (20*da.dahinc) THEN            02359
                   newvalue _ da.daleft + (20*da.dahinc)                 02360
                END                                                      02375
             END                                                         02374
          END                                                            02366
       END                                                               02390
    ELSE %horizontal boundry%                                            0915
       BEGIN                                                             02377
       IF da.dabottom = value THEN                                       0916
          BEGIN                                                          02379
          IF allowdelete THEN                                            02361
             BEGIN                                                       02380
             IF newvalue < da.datop THEN                                 0917
                newvalue _ da.datop                                      0918
             END                                                         02382
          ELSE                                                           02362
             BEGIN                                                       02381
             IF newvalue < da.datop + (2*da.davinc) THEN                 02363
                newvalue _ da.datop + (2*da.davinc)                      02364
             END                                                         02383
          END                                                            02378
       ELSE                                                              0920
          BEGIN                                                          02386
          IF da.datop = value THEN                                       0921
             BEGIN                                                       02388
             IF allowdelete THEN                                         02392
                BEGIN                                                    02399
                IF newvalue > da.dabottom THEN                           0922
                   newvalue _ da.dabottom                                0923
                END                                                      02398
             ELSE                                                        02393
                BEGIN                                                    02397
                IF newvalue > da.dabottom - (2*da.davinc) THEN
                                                                         02394
                   newvalue _ da.dabottom - (2*da.davinc)                02395
                END                                                      02396
             END                                                         02385
```

```
            END                                                    02387
         END                                                       02384
   UNTIL (&da _ &da + dal) = end;                                   0926
   &da _ $dpyarea;                                                02325
   DO IF da.daexis THEN                                           02326
      IF type THEN %vertical boundry%                             02327
         BEGIN                                                    02400
         IF da.daleft = value THEN da.daleft _ newvalue           02328
         ELSE IF da.daright = value THEN da.daright _ newvalue    02333
         END                                                      02401
      ELSE %horizontal boundry%                                   02339
         BEGIN                                                    02402
         IF da.dabottom = value THEN da.dabottom _ newvalue       02340
         ELSE IF da.datop = value THEN da.datop _ newvalue        02344
         END                                                      02403
   UNTIL (&da _ &da + dal) = end;                                 02350
   RETURN;                                                          0991
   END.                                                             0992
                                                                    0993
(fixBuf) %reapportion buffer space among all windows%
PROCEDURE;                                                          0994
   %reapportion lsrt space amoung all of the da's in the display area
   record%                                                          0995
   %-------------%                                                  0996
   LOCAL end, da, lines, columns, rt, entry, height, width, db;     0997
   REF da, entry;                                                   0998
   %Note:  This version of fixbuf sets max rows to bottom of da and
   max columns to far right%
                                                                    0999
   IF dacnt NOT IN [1,damax] THEN                                  01000
      err($"display area error");                                  01001
   end _ (&da _ $dpyarea) + dacnt*dal;                             01002
   DO  %check for windows to be deleted%                           01836
      IF da.daexis THEN                                            01837
         BEGIN                                                     01838
         height _ da.dabottom - da.datop;                          01839
         lines _ height/da.davinc;                                 01840
         width _ da.daright-da.daleft;                             01841
         columns _ width/da.dahinc;                                01842
         IF height <= 0 OR lines < 2 OR width <= 0 OR columns < 20
         THEN                                                      01843
            delda(&da);                                            01853
         END                                                       01866
   UNTIL (&da _ &da + dal) = end;                                  01867
   &da _ $dpyarea;                                                 01868
   rt _ 1;                                                         01003
   DO                                                              01004
      IF da.daexis THEN                                            01005
         BEGIN                                                     01006
         height _ da.dabottom - da.datop;                          01007
         lines _ height/da.davinc;                                 01008
         width _ da.daright-da.daleft;                             01009
         columns _ width/da.dahinc;                                01011
         IF da.darneighbor THEN columns _ columns-3;               01027
         IF udpcolmax IN (0, columns) OR (udpwrapcol AND udpwrapcol <
         columns) THEN columns _ udpcolmax;                        02636
```

```
         IF da.dabneighbor THEN lines _ lines-1;                    01796
         %reapportion display list entries%                         01022
            freelsrt($dspblk, &da);                                 01023
            IF NOT maklsrt($dspblk, lines + 2, &da) THEN err($"NLS
            out of space for LSRT");                                01024
         da.damcol _ columns*da.dahinc;                             01026
         da.damrow _ lines*da.davinc;                               01028
         da.damind _ MIN((da.damcol - da.daind)/da.daind*da.daind,
         tamind*da.daind);                                          02407
         %deallocate and reallocate da%                             01771
            dealocda(&da);                                          01772
            <NLS, DSPGEN, alocda>(&da);                             01773
         END                                                        01046
   UNTIL rt >= rtlast OR (&da _ &da + dal) = end;                   01047
   IF &da < end THEN                                                01048
      DO                                                            01049
         IF da.daexis THEN                                          01050
            delda(&da)                                              01051
      UNTIL (&da _ &da + dal) >= end;                               01052
   %update rtfree, next free "entry"%                               01053
      rtfree _ rt;                                                  01054
   RETURN;                                                          01055
   END.                                                             01056
                                                                    01057
(lctile) %find the file represented in the window containing the
cursor during the last input control char%
PROCEDURE;                                                          01058
   %returns the file number of the csp of the display area          01059
   containing the cursor when the last character was input%         01060
   %---------------%                                               01061
   LOCAL stid;                                                      01062
   stid _ lccsp();                                                  01063
   RETURN(stid.stfile);                                             01064
   END.                                                             01065
                                                                    01066
(lccsp) %find Current Statement Pointer (CSP) for the window
containing the cursor during the last input control char%
PROCEDURE;                                                          01067
   %returns the csp of the display area containing the cursor       01068
   when the last character was input%                               01069
   %---------------%                                               01070
   LOCAL da;                                                        01071
   REF da;                                                          01072
   IF NOT (&da _ dsparea(lcda())) THEN                              01073
      err($"lcda failed, lccsp");                                   01074
   RETURN(da.dacsp, da.daccnt);                                     01075
   END.                                                             01076
                                                                    01077
(lcda) %find the window containing the cursor during the last input
control character%
PROCEDURE;                                                          01078
   LOCAL dacords;                                                   01079
   IF nlmode = typewriter THEN RETURN((&tda-$dpyarea)/dal + 1, 0);
                                                                    01080
   dacords _ lccords();                                             01081
   RETURN(findda(dacords : dacords.xcord, dacords.ycord), dacords);
```

```
                                                                    01082
   END.                                                             01083
                                                                    01084
(lda)  % returns address of Display Area descriptor for the display
area where the cursor was the Last time a character was input %
PROCEDURE;                                                          01241
   RETURN (dsparea(lcda())); END.
                                                                    01242
(lccords) %get coordinates associated with last control character
input%
PROCEDURE;                                                          01085
   LOCAL coords;                                                    01086
   coords _ 0;                                                      01087
   IF inpjfn THEN                                                   01088
      BEGIN %read coords from control file%                         01089
      r1 _ inpjfn;                                                  01090
      !JSYS bin;                                                    01091
      coords.xcord.xc1 _ r2;                                        01092
      !JSYS bin;                                                    01093
      coords.xcord.xc2 _ r2;                                        01094
      !JSYS bin;                                                    01095
      coords.ycord.yc1 _ r2;                                        01096
      !JSYS bin;                                                    01097
      coords.ycord.yc2 _ r2;                                        01098
      END                                                           01099
   ELSE %use the fast jsys%                                         01100
      IF tenex < 13200 THEN                                         02272
         BEGIN                                                      01101
         !JSYS 407B; %read coords of last character%                01102
         !MOVSS r1;                                                 01103
         coords _ r1;                                               01104
         END                                                        01105
      ELSE coords _ gcords;                                         02273
   IF inpwatchjfn AND (NOT recrunfil) THEN                          01106
      BEGIN %write coords to control file%                          01107
      r1 _ inpwatchjfn;                                             01108
      r2 _ coords.xcord.xc1;                                        01109
      !JSYS bout;                                                   01110
      r2 _ coords.xcord.xc2;                                        01111
      !JSYS bout;                                                   01112
      r2 _ coords.ycord.yc1;                                        01113
      !JSYS bout;                                                   01114
      r2 _ coords.ycord.yc2;                                        01115
      !JSYS bout;                                                   01116
      END;                                                          01117
   CASE nldevice OF                                                 01118
      = devlproc:                                                   01119
         BEGIN                                                      01120
         coords.ycord _ lpymax - coords.ycord; %y-cord%             01121
         END;                                                       01122
      ENDCASE                                                       01123
         BEGIN                                                      01124
         coords.xcord _ coords.xcord * 256; %x-cord%                01125
         coords.ycord _                                             01126
            (bmarg-tmarg) - coords.ycord * 256; %y-cord%            01127
         END;                                                       01128
```

```
   RETURN(coords);                                                    01129
   END.                                                               01130
                                                                      01131
(findda) %find the window which contains the passed ccordinates%
PROCEDURE (dacords);                                                  01132
   %do not round %                                                    01133
   LOCAL x, y, da, end, bottom, right;                                01134
   REF da;                                                            01135
   IF dacnt NOT IN [1, damax] THEN                                    01136
      err($"Illegal display area");                                   01137
   x _ MAX(MIN(taright, dacords.xcord), taleft);                      01138
   y _ MAX(MIN(tabottom, dacords.ycord), tatop);                      01139
   end _ (&da _ $dpyarea) + dacnt*dal;                                01140
   DO                                                                 01141
      IF da.daexis AND NOT da.daseq AND NOT da.daauxiliary THEN       01142
         BEGIN                                                        02412
         right _ da.daright;                                          02414
         IF da.darneighbor THEN right _ right-1;                      02415
         bottom _ da.dabottom;                                        02416
         IF da.dabneighbor THEN bottom _ bottom-1;                    02417
         IF x IN [da.daleft, right]
         AND y IN [da.datop, bottom] THEN                             02411
            RETURN((&da-$dpyarea)/dal+1, x, y);                       01143
         END                                                          02413
   UNTIL (&da _ &da + dal) = end;                                     01144
   err($"No valid File windows found in findda");                     01145
   END.                                                               01146
                                                                      01147
(dsparea) %get window record address from window number -- returns
zero if window is not allocated%
PROCEDURE (dano);                                                     01148
   LOCAL entry;                                                       01149
   IF dano NOT IN [1, damax] THEN                                     01150
      err($"Illegal window identifier encountered in DSPAREA");       01760
   entry _ $dpyarea +(dano-1)*dal;                                    01151
   IF NOT [entry].daexis THEN                                         01152
      err($"Illegal window identifier encountered in DSPAREA");       02538
   RETURN(entry);                                                     01153
   END.                                                               01154
                                                                      01155
(delda) %delete window%
PROCEDURE (da);                                                       01156
   %does not reappropriate buffer space for the da--just deletes
                                                                      01157
   da from display area records and tss--fixbuf must be called to
                                                                      01158
   reappropriate the buffer space%                                    01159
   %-------------%                                                    01160
   LOCAL end, lfrozen, left, right, top, bottom;                      01161
   REF da, lfrozen, left, right, top, bottom;                         01162
   %reset neighbor pointers of this da's neighbors%                   02352
      IF da.dalneighbor THEN %fix left neighbor's right-neighbor
      pointer%                                                        01845
         LOOP                                                         02441
            BEGIN                                                     02442
            &left _ dsparea(da.dalneighbor);                          02434
```

```
        %see if top neighbors are LEFT's right neighbor's also%
                                                                02527
           &top _ &da;                                          02435
           WHILE top.datneighbor DO                             02436
              BEGIN                                             02437
              &top _ dsparea(top.datneighbor);                  02539
              IF top.datop IN [left.datop, left.dabottom) OR
              top.dabottom IN (left.datop, left.dabottom] THEN
                                                                02438
                 BEGIN                                          02444
                 left.darneighbor _ (&top-$dpyarea)/dal+1;      02440
                 EXIT LOOP 2;                                   02446
                 END;                                           02445
              END;                                              02439
        %see if bottom neighbors are LEFT's right neighbor's
        also%                                                   02528
           &bottom _ &da;                                       02455
           WHILE bottom.dabneighbor DO                          02447
              BEGIN                                             02448
              &bottom _ dsparea(bottom.dabneighbor);            02540
              IF bottom.datop IN [left.datop, left.dabottom) OR
              bottom.dabottom IN (left.datop, left.dabottom] THEN
                                                                02449
                 BEGIN                                          02450
                 left.darneighbor _ (&bottom-$dpyarea)/dal+1;
                                                                02451
                 EXIT LOOP 2;                                   02452
                 END;                                           02453
              END;                                              02454
        left.darneighbor _ da.darneighbor;                      02456
        EXIT LOOP;                                              02529
        END;                                                    02443
     IF da.darneighbor THEN %fix right neighbor's left-neighbor
     pointer%                                                   01847
        LOOP                                                    02501
           BEGIN                                                02457
           &right _ dsparea(da.darneighbor);                    02458
           %see if top neighbors are RIGHT's left neighbor's also%
                                                                02530
              &top _ &da;                                       02459
              WHILE top.da.datneighbor DO                       02460
                 BEGIN                                          02461
                 &top _ dsparea(top.da.datneighbor);            02541
                 IF top.datop IN [right.datop, right.dabottom) OR
                 top.dabottom IN (right.datop, right.dabottom] THEN
                                                                02462
                    BEGIN                                       02463
                    right.dalneighbor _ (&top-$dpyarea)/dal+1;  02464
                    EXIT LOOP 2;                                02465
                    END;                                        02466
                 END;                                           02467
           %see if bottom neighbors are LEFT's right neighbor's
           also%                                                02533
              &bottom _ &da;                                    02468
              WHILE bottom.da.dabneighbor DO                    02469
                 BEGIN                                          02470
```

```
                &bottom _ dsparea(bottom.da.dabneighbor);      02542
                IF bottom.datop IN [right.datop, right.dabottom) OR
                bottom.dabottom IN (right.datop, right.dabottom]
                THEN                                           02471
                   BEGIN                                       02472
                   right.dalneighbor _ (&bottom-$dpyarea)/dal+1;
                                                               02473
                   EXIT LOOP 2;                                02474
                   END;                                        02475
                END;                                           02476
             right.dalneighbor _ da.dalneighbor;               02477
             EXIT LOOP;                                        02531
             END;                                              02478
          IF da.datneighbor THEN %fix top neighbor's bottom-neighbor
          pointer%                                             01849
             LOOP                                              02502
                BEGIN                                          02479
                &top _ dsparea(da.datneighbor);                02480
                %see if left neighbors are TOP's bottom neighbors also%
                                                               02532
                   &left _ &da;                                02481
                   WHILE left.dalneighbor DO                   02482
                      BEGIN                                    02483
                      &left _ dsparea(left.dalneighbor);       02543
                      IF left.daleft IN [top.daleft, top.daright) OR
                      left.daright IN (top.daleft, top.daright] THEN
                                                               02484
                         BEGIN                                 02485
                         top.dabneighbor _ (&left-$dpyarea)/dal+1; 02486
                         EXIT LOOP 2;                          02487
                         END;                                  02488
                      END;                                     02489
                %see if right neighbors are TOP's bottom neighbors also%
                                                               02534
                   &right _ &da;                               02490
                   WHILE right.darneighbor DO                  02491
                      BEGIN                                    02492
                      &right _ dsparea(right.darneighbor);     02544
                      IF right.daleft IN [top.daleft, top.daright) OR
                      right.daright IN (top.daleft, top.daright] THEN
                                                               02493
                         BEGIN                                 02494
                         top.dabneighbor _ (&right-$dpyarea)/dal+1; 02495
                         EXIT LOOP 2;                          02496
                         END;                                  02497
                      END;                                     02498
                top.dabneighbor _ da.dabneighbor;              02499
                EXIT LOOP;                                     02526
                END;                                           02500
          IF da.dabneighbor THEN %fix bottom neighbor's top-neighbor
          pointer%                                             01851
             LOOP                                              02503
                BEGIN                                          02504
                &bottom _ dsparea(da.dabneighbor);             02505
                %see if left neighbors are BOTTOM's top neighbors also%
                                                               02535
```

```
                &left _ &da;                                            02506
                WHILE left.dalneighbor DO                               02507
                   BEGIN                                                02508
                   &left _ dsparea(left.dalneighbor);                   02545
                   IF left.daleft IN [bottom.daleft, bottom.daright)
                   OR left.daright IN (bottom.daleft, bottom.daright]
                   THEN                                                 02509
                      BEGIN                                             02510
                      bottom.datneighbor _ (&left-$dpyarea)/dal+1;
                                                                        02511
                      EXIT LOOP 2;                                      02512
                      END;                                              02513
                   END;                                                 02514
             %see if right neighbors are BOTTOM's top neighbors also%
                                                                        02536
                &right _ &da;                                           02515
                WHILE right.darneighbor DO                              02516
                   BEGIN                                                02517
                   &right _ dsparea(right.darneighbor);                 02546
                   IF right.daleft IN [bottom.daleft, bottom.daright)
                   OR right.daright IN (bottom.daleft, bottom.daright]
                   THEN                                                 02518
                      BEGIN                                             02519
                      bottom.datneighbor _ (&right-$dpyarea)/dal+1;
                                                                        02520
                      EXIT LOOP 2;                                      02521
                      END;                                              02522
                   END;                                                 02523
             bottom.datneighbor _ da.datneighbor;                       02524
             EXIT LOOP;                                                 02537
             END;                                                       02525
   %put frzlst entries on free list%                                    01163
      IF &lfrozen _ da.dafrzl THEN                                      01164
         BEGIN                                                          01165
         WHILE &lfrozen DO                                              01166
            BEGIN                                                       01167
            lfrozen.fzexis _ FALSE;                                     01168
            &lfrozen _ lfrozen.fznext;                                  01169
            END;                                                        01170
         lfrozen.fznext := fzfree := da.dafrzl := 0;                    01171
         END;                                                           01172
   %release link stack%                                                 01173
      freefrring(da.dalink := 0);                                       01174
   %tell tss system to delete da%                                       01774
      IF da.dahandle THEN dealocda(&da);                                01775
   %deallocate core associated with LSRT%                               01188
      freelsrt ($dspblk, &da);                                          01189
   %clear out da da%                                                    01190
      end _ &da + dal;                                                  01191
      DO da _ 0 UNTIL (&da _ &da + 1) = end;                            01192
   RETURN;                                                              01193
   END.                                                                 01194
                                                                        01195
(newda) %allocate new window%
PROCEDURE;                                                              01196
   LOCAL entry, end;                                                    01197
```

```
    REF entry;                                                          01198
    IF dacnt NOT IN [0, damax] THEN                                     01199
       err($"Illegal dacnt, newda");                                    01200
    end _ (&entry _ $dpyarea-dal) + dacnt*dal;                          01201
    UNTIL (&entry _ &entry + dal) > end DO                              01202
       IF NOT entry.daexis THEN EXIT;                                   01203
    IF &entry > end THEN                                                01204
       BEGIN                                                            01205
       IF dacnt = damax THEN                                            01206
          err($"Too many display areas");                               01207
       dacnt _ dacnt + 1;                                               01208
       END;                                                             01209
    entry.daexis _ TRUE;                                                01210
    entry.dafrzl _ 0;                                                   01211
    entry.dalink _ newfrring(frrsize);  %get return rings%              01212
    entry.dacsp _ endfil;                                               01213
    entry.daccnt _ 1;                                                   01214
    entry.daempty _ TRUE;                                               01215
    entry.daauxiliary _ FALSE; %assume to be used for display
    purposes%                                                           01216
    entry.dalsrt _ 0; %no space allocated yet%                          01217
    entry.dalsidb _ ((&entry - $dpyarea)/dal )* lsbtsize +
    $lsbitables;                                                        01776
    dassnbit(entry.dalsidb, -1, lsbtsize);                              01777
    RETURN(&entry);                                                     01218
    END.                                                                01219
                                                                        01220
(copyda) %copy contents of da excluding file return ring%
PROCEDURE (dafrom, dato);                                               01221
    LOCAL end, dadest, frr, save;                                       01222
    REF dafrom, dato, dadest, frr;                                      01223
    &dadest _ &dato;  %save original value%                             01224
    save _ dato.dalsidb;                                                01778
    &frr _ dato.dalink;                                                 01225
    %copy da entry%                                                     01226
       end _ &dafrom + dal;                                             01227
       DO                                                               01228
          BEGIN                                                         01229
          dato _ dafrom;                                                01230
          BUMP &dato;                                                   01231
          END                                                           01232
       UNTIL (&dafrom _ &dafrom + 1) = end;                             01233
    dadest.dalink _ &frr;                                               01234
    dadest.dalsrt _ FALSE;  %new da - no LSRT allocated%                01235
    dadest.dashrinkcnt _ FALSE;                                         01236
    dadest.dafrozen _ FALSE;                                            01237
    dadest.dalsidb _ save;                                              01779
    FOR end _ dafrom.dalsid UP UNTIL >= dafrom.dalsid +lsbtsize DO
                                                                        01780
       BEGIN                                                            01781
       [save] _ [end]; %copy lsid bit table%                            01782
       BUMP save;                                                       01783
       END;                                                             01784
    RETURN;                                                             01238
    END.                                                                01239
                                                                        01240
```

```
%.....return ring routines.....%                                        01243
   (newfrring)  % allocate a new file return ring %
    PROCEDURE (size);                                                   01245
      LOCAL frr;                                                        01246
      REF frr;                                                          01247
      IF size NOT IN [1,frrmax] THEN err($"Illegal size requested for
      file return ring");                                               01248
      %get frr block%                                                   01249
         &frr _ getblk((size*frrelen)+frrhlen, $dspblk) + bhl;          01250
         IF &frr <= bhl THEN err($"Insufficient space for new file
         return ring");                                                 01251
      %init frr block%                                                  01252
         frr.frhlast _ size-1;                                          01253
         frr.frhexis _ TRUE;                                            01254
      RETURN(&frr);                                                     01255
      END.                                                              01256
   (freefrring)  %free file return ring%
   PROCEDURE (frh);                                                     01257
      LOCAL i, end, frr;                                                01258
      REF frr, frh;                                                     01259
      IF NOT frh.frhexis THEN err($"Illegal file return ring");         01260
      end _ &frh + frrhlen + (frrelen*frh.frhlast);                     01261
      %free statement return rings%                                     01262
         FOR &frr _ &frh +frrhlen UP frrelen UNTIL > end DO             01263
            IF frr.frexis THEN freesrring(frr.frsrring);                01264
      end _ end + frrelen-1;                                            01265
      %zero block as a precaution%                                      01266
         FOR i _ &frh UP UNTIL > end DO [i] _ 0;                        01267
      %free block%                                                      01268
         freeblk(&frh-bhl, $dspblk);                                    01269
      RETURN;                                                           01270
      END.                                                              01271
   (pushfrring)  %push file return ring%
   PROCEDURE (frr, filename, fileno);                                   01272
      LOCAL srr, fre;                                                   01273
      REF frr, fre, srr, filename;                                      01274
      IF NOT frr.frhexis THEN err($"Illegal file return ring");         01275
      % zero file number of current top SRRING %                       01276
         &fre _ &frr + frrhlen + (frrelen*frr.frhtop);                  01277
         IF fre.frexis THEN                                             01278
            BEGIN                                                       01279
            &srr _ fre.frsrring;                                        01280
            srr.srhfileno _ 0;                                          01281
            END;                                                        01282
      %increment top-of-ring pointer%                                   01283
         frr.frhtop _ IF frr.frhtop >= frr.frhlast THEN 0 ELSE
         frr.frhtop+1;                                                  01284
      %get address of new top entry%                                    01285
         &fre _ &frr + frrhlen + (frrelen*frr.frhtop);                  01286
      %if there is an old statement return ring, free it%               01287
         IF fre.frexis AND fre.frsrring THEN freesrring(fre.frsrring);
                                                                        01288
      %allocate a statement return ring%                                01289
         fre.frsrring _ &srr _ newsrring(srrsize);                      01290
         fre.frexis _ TRUE;                                             01291
      %allocate a string for file name%                                 01292
```

```
      srr.srhfname _ getstring(filename.L, $dspblk);                    01293
   %store away file name and number%                                    01294
      *[srr.srhfname]* _ *filename*;                                    01295
      srr.srhfileno _ fileno;                                           01296
   RETURN;                                                              01297
   END.                                                                 01298
(readfrring)  %Read file return ring entry.%
PROCEDURE (frr, index);                                                 01299
   % case index of                                                      01300
      = 0: read top entry                                               01301
      > 0 decrement over index good entries and return contents         01302
      endcase;%                                                         01303
   %returns address of file name string (READ ONLY) and address of
   statement return ring%                                               01304
   % ------------------- %                                             01305
   LOCAL i, stid, j, last, base, count, fre, srr;                       01306
   REF frr, fre, srr;                                                   01307
   IF NOT frr.frhexis THEN err($"Illegal file return ring");            01308
   %initialization%                                                     01309
      j _ frr.frhtop;                                                   01310
      last _ frr.frhlast;                                               01311
      base _ &frr + frrhlen;                                            01312
      IF index < 0 THEN                                                 01313
         err($"Illegal index in readfrring");                           01314
      count _ 0;  %used to detect empty ring%                           01315
      &fre _ base + (frrelen*j);  %current top%                         01316
   FOR i _ index MOD (last+1) DOWN UNTIL <= 0 DO                        01317
      BEGIN                                                             01318
      %move one entry%                                                  01319
         j _ IF j <= 0 THEN last ELSE j -1;                             01320
         &fre _ base + (frrelen*j);                                     01321
      %continue moving until find valid entry%                          01322
         UNTIL fre.frexis DO                                            01323
            BEGIN                                                       01324
            j _ IF j <= 0 THEN last ELSE j -1;                          01325
            &fre _ base + (frrelen*j);                                  01326
            IF (count _ count+1) > last THEN                            01327
               err($"no entries in statement return ring --
               backfrring");                                            01328
            END;                                                        01329
         count _ 0;                                                     01330
      END;                                                              01331
   IF NOT fre.frexis THEN err($"current entry empty in readfrring");
                                                                        01332
   &srr _ fre.frsrring;                                                 01333
   IF NOT srr.srhexis THEN                                              02408
      err($"Illegal statemment return ring detected in readfrring");
                                                                        02409
   RETURN(srr.srhfname, &srr);                                          01334
   END.                                                                 01335
                                                                        01336
(trrlength)  %return number of entries in the file return ring.%
PROCEDURE (frr);                                                        01337
   LOCAL i, j, k, last, base, count, fre, srr, top;                     01338
   REF frr, fre, srr;                                                   01339
   % ------------------- %                                             01340
```

```
   IF NOT frr.frhexis THEN                                              01341
      err($"Illegal file return ring in frrlength");                    01342
   %initialization%                                                     01343
      j _ top _ frr.frhtop;                                             01344
      last _ frr.frhlast;                                               01345
      base _ &frr + frrhlen;                                            01346
      &fre _ base + (frrelen*top);  %current top%                       01347
      IF fre.frexis THEN k _ 0 ELSE k _ 1;  %see end of proc%           02817
      count _ 0;                                                        01348
      UNTIL fre.frexis DO                                               01349
         BEGIN                                                          01350
         j _ IF j <= 0 THEN last ELSE j -1;                             01351
         &fre _ base + (frrelen*j);                                     01352
         IF (count _ count+1) > last OR j = top THEN                    01353
            RETURN(-1);                                                 01354
         END;                                                           01355
   FOR i _ 0 UP UNTIL > last DO                                         01356
      BEGIN                                                             01357
      %move one entry%                                                  01358
         j _ IF j <= 0 THEN last ELSE j -1;                             01359
         &fre _ base + (frrelen*j);                                     01360
         IF j = top THEN EXIT LOOP;                                     01361
      %continue moving until find valid entry%                          01362
         count _ 0;                                                     01363
         UNTIL fre.frexis DO                                            01364
            BEGIN                                                       01365
            j _ IF j <= 0 THEN last ELSE j -1;                          01366
            &fre _ base + (frrelen*j);                                  01367
            IF j = top THEN EXIT LOOP 2;                                01368
            IF (count _ count+1) > last THEN                            01369
               RETURN(-1);                                              01370
            END;                                                        01371
      END;                                                              01372
   i _ i + k;  %to take account of the different way readfrring acts
   if the top entry is empty; need the count of entries to include
   that one%                                                            02818
   RETURN(i);                                                           01373
   END.                                                                 01374
                                                                        01375
(uplsfnm) %change file name ofn to nfn in all file return rings%
PROCEDURE (ofn, nfn);                                                   01376
   %ofn,   address of string containing complete old file name%         01377
   %nfn,   address of string containing complete new file name%         01378
   LOCAL da, end, i, str, len, srr;                                     01379
   REF ofn, nfn, da, str, srr;                                          01380
   %-----------------------%                                            01381
   end _ $dpyarea + dacnt*dal;                                          01382
   FOR &da _ $dpyarea UP dal UNTIL > end DO                             01383
      IF da.daexis THEN                                                 01384
         BEGIN                                                          01808
         len _ frrlength(da.dalink);                                    01810
         FOR i _ 0 UP UNTIL > len DO                                    01385
            BEGIN                                                       01799
            ON SIGNAL ELSE REPEAT LOOP;                                 01813
            &str _ readfrring(da.dalink, i: &srr);                      01812
            ON SIGNAL ELSE;                                             01814
```

```
            IF &str AND (*str* = *ofn*) THEN                          01386
               BEGIN                                                  01387
               IF str.M < nfn.L THEN                                  01388
                  BEGIN                                               01389
                  freestring(&str, $dspblk);                          01390
                  &str _ getstring(nfn.L, $dspblk);                   01391
                  srr.srhfname _ &str;                                02808
                  END;                                                01392
               *str* _ *nfn*;                                         01393
               END;                                                   01394
            END;                                                      01800
         END;                                                         01809
   RETURN;                                                            01395
   END.
                                                                      01396
(mkdelfrr) PROCEDURE (frr, filename);                                 02791
   %Mark file return ring entries for deleted files.  Set frexis
   field to FALSE, and free associated statement return ring%         02792
   LOCAL fre, srr, frrfname;                                          02793
   REF frr, fre, srr, frrfname, filename;                             02794
   FOR &fre _ &frr + frrhlen UP UNTIL > &frr + frr.frhlast + frrhlen
   DO                                                                 02795
      IF fre.frexis THEN                                              02796
         BEGIN                                                        02797
         &srr _ fre.frsrring;                                         02798
         &frrfname _ srr.srhfname;                                    02799
         IF *frrfname* = *filename* THEN                              02800
            BEGIN                                                     02801
            fre.frexis _ FALSE;                                       02802
            freestring(&srr, dspblk);                                 02803
            END;                                                      02804
         END;                                                         02805
   RETURN;                                                            02806
   END.                                                               02807
(newsrring)  %allocate a new statement return ring%
PROCEDURE (size);                                                     01397
   LOCAL srr;                                                         01398
   REF srr;                                                           01399
   IF size NOT IN [1,srrmax] THEN err($"Illegal size requested for
   statement return ring");                                           01400
   %get srr block%                                                    01401
      &srr _ getblk((size*srrelen)+srrhlen, $dspblk) + bhl;           01402
      IF &srr <= bhl THEN                                             01403
         err($"Insufficient space for new statement return ring");
                                                                      01404
   %init srr block%                                                   01405
      srr.srhlast _ size-1;                                           01406
      srr.srhexis _ TRUE;                                             01407
   RETURN(&srr);                                                      01408
   END.                                                               01409
(freesrring)  %free statement return ring%
PROCEDURE (srh);                                                      01410
   LOCAL i, end;                                                      01411
   REF srh;                                                           01412
   IF NOT srh.srhexis THEN err($"Illegal statement return ring");
                                                                      01413
```

```
   %free file name string%                                              01414
      IF srh.srhfname THEN freestring(srh.srhfname, $dspblk);           01415
      srh.srhfname _ 0;                                                 01416
   %zero block as a precaution%                                         01417
      end _ &srh + srrhlen + (srh.srhlast*srrelen) + srrelen-1;         01418
      FOR i _ &srh UP UNTIL > end DO [i] _ 0;                           01419
   %free block%                                                         01420
      freeblk(&srh-bhl, $dspblk);                                       01421
   RETURN;                                                              01422
   END.                                                                 01423
(copysrring)  %copy statement return ring%
PROCEDURE (fromsrr, tosrr);                                             01424
   %does not change file name string or file number in tosrr.  Copies
   body of fromsrr to tosrr and set top-of-ring in tosrr to
   correspond with fromsrr.%                                            01425
   LOCAL i, end;                                                        01426
   REF fromsrr, tosrr;                                                  01427
   IF NOT fromsrr.srhexis OR NOT tosrr.srhexis THEN                     01428
      err($"Illegal statement return ring in copysrring");              01429
   %copy block%                                                         01430
      end _ srrhlen + (tosrr.srhlast*srrelen) + srrelen-1;              01431
      FOR i _ srrhlen UP UNTIL > end DO tosrr[i] _ fromsrr[i];          01432
      tosrr.srhtop _ fromsrr.srhtop;                                    01433
   RETURN;                                                              01434
   END.                                                                 01435
(pushsrring)  %push contents onto statement return ring%
PROCEDURE (srr, stid, cc, vs1, vs2);                                    01436
   %increment and store%                                                01437
   LOCAL sre; REF sre;                                                  01438
   REF srr;                                                             01439
   %verify file number%                                                 01440
      IF srr.srhfileno AND stid.stfile NOT= srr.srhfileno THEN          01441
         err($"file numbers do not match in pushsrring");               01442
   %increment top-of-ring pointer%                                      01443
      srr.srhtop _ IF srr.srhtop >= srr.srhlast THEN 0 ELSE
      srr.srhtop+1;                                                     01444
   %get address of top entry%                                           01445
      &sre _ &srr + srrhlen + (srrelen*srr.srhtop);                     01446
   %store new values%                                                   01447
      sre.srpsid _ stid.stpsid;                                         01448
      sre.srcc _ cc;                                                    01449
      sre.srvs1 _ vs1;                                                  01450
      sre.srvs2 _ vs2;                                                  01451
      sre.srexis _ TRUE;                                                01452
   RETURN;                                                              01453
   END.                                                                 01454
(storesrring)  %store contents onto statement return ring.  INDEX has
the same semantics as in readsrring.%
PROCEDURE (srr, index, stid, cc, vs1, vs2);                             02745
   LOCAL sre, i, j, last, base, count;                                  02746
   REF srr, sre;                                                        02747
   %verify file number%                                                 02748
      IF srr.srhfileno AND stid.stfile NOT= srr.srhfileno THEN          02749
         BEGIN                                                          02750
         dismes(1, $"Bad file number in stmt return ring; advise you
         ^C, reset, and call NLS.");                                    02751
```

```
      RETURN;                                                    02752
      END;                                                       02753
   CASE index OF   % pick out the srr entry to update %         02754
      %> srr: &sre _ index;  caller specified entry%            02755
      = 0:  &sre _ &srr + srrhlen + srrelen*srr.srhtop;  %top%  02756
      ENDCASE   %use index to entry as in readsrring%           02757
         BEGIN                                                   02758
         j _ srr.srhtop;                                         02759
         last _ srr.srhlast;                                     02760
         base _ &srr + srrhlen;                                  02761
         count _ 0;  %used to detect empty ring%                 02762
         IF index < 0 THEN err($"Illegal index in storesrring");
                                                                 02763
         &sre _ base + (srrelen*j);                              02764
         FOR i _ index MOD (last+1) DOWN UNTIL <= 0 DO           02765
            BEGIN                                                02766
            %move one entry%                                     02767
               j _ IF j <= 0 THEN last ELSE j -1;                02768
               &sre _ base + (srrelen*j);                        02769
            %continue backing up until find valid entry%         02770
               UNTIL sre.srexis DO                               02771
                  BEGIN                                          02772
                  j _ IF j <= 0 THEN last ELSE j -1;             02773
                  &sre _ base + (srrelen*j);                     02774
                  IF (count _ count+1) > last THEN               02775
                     err($"no entries in statement return ring --
                     storesrring");                              02776
                  END;                                           02777
               count _ 0;                                        02778
            END;                                                 02779
         IF NOT sre.srexis THEN                                  02780
            err($"current entry empty in storesrring");          02781
         END;                                                    02782
   %store new values%                                            02783
      sre.srpsid _ stid.stpsid;                                  02784
      sre.srcc _ cc;                                             02785
      sre.srvs1 _ vs1;                                           02786
      sre.srvs2 _ vs2;                                           02787
      sre.srexis _ TRUE;                                         02788
   RETURN;                                                       02789
   END.                                                          02790
(readsrring)  %Read statement return ring entry.%                01471
   PROCEDURE (srr, index);                                       01759
   % case index of                                              01472
      = 0: read top entry                                        01473
      > 0: decrement over index good entries and return contents
                                                                 01474
      endcase;%                                                  01475
   %returns stid, cc, vs1, vs2%                                  01476
   % ------------------- %                                       01477
   LOCAL i, stid, j, last, base, count, inc, sre;                01478
   REF srr, sre;                                                 01479
   IF NOT srr.srhexis THEN err($"Illegal statement return ring");
                                                                 01480
   %initialization%                                              01481
      stid _ 0;                                                  02683
```

```
      stid.stfile _ srr.srhfileno;                                    01482
      j _ srr.srhtop;                                                 01483
      last _ srr.srhlast;                                             01484
      base _ &srr + srrhlen;                                          01485
      count _ 0;  %used to detect empty ring%                         01486
      IF index < 0 THEN err($"Illegal index in readsrring");          01487
      &sre _ base + (srrelen*j);                                      01488
   %move through the ring%                                            01489
      FOR i _ index MOD (last+1) DOWN UNTIL <= 0 DO                   01490
         BEGIN                                                        01491
         %move one entry%                                             01492
            j _ IF j <= 0 THEN last ELSE j -1;                        01493
            &sre _ base + (srrelen*j);                                01494
         %continue backing up until find valid entry%                 01495
            UNTIL sre.srexis DO                                       01496
               BEGIN                                                  01497
               j _ IF j <= 0 THEN last ELSE j -1;                     01498
               &sre _ base + (srrelen*j);                             01499
               IF (count _ count+1) > last THEN                       01500
                  err($"no entries in statement return ring --
                  readsrring");                                       01501
               END;                                                   01502
            count _ 0;                                                01503
         END;                                                         01504
   IF NOT sre.srexis THEN                                             01505
      err($"current entry empty in readsrring");                      01506
   stid.stpsid _ sre.srpsid;                                          01507
   RETURN(stid, sre.srcc, sre.srvs1, sre.srvs2);                      01508
   END.                                                               01509
                                                                      01510
(srrlength)  %return number of entries in statement return ring.%
PROCEDURE (srr);                                                      01511
   LOCAL i, j, last, base, count, sre, top;                           01512
   REF srr, sre;                                                      01513
   % ------------------- %                                            01514
   IF NOT srr.srhexis THEN                                            01515
      err($"Illegal statement return ring in srrlength");             01516
   %initialization%                                                   01517
      j _ top _ srr.srhtop;                                           01518
      last _ srr.srhlast;                                             01519
      base _ &srr + srrhlen;                                          01520
      &sre _ base + (srrelen*top);                                    01521
      count _ 0;                                                      01522
      UNTIL sre.srexis DO                                             01523
         BEGIN                                                        01524
         j _ IF j <= 0 THEN last ELSE j -1;                           01525
         &sre _ base + (srrelen*j);                                   01526
         IF (count _ count+1) > last OR j = top THEN                  01527
            RETURN(0);                                                01528
         END;                                                         01529
   %count valid entries in the ring%                                  01530
      FOR i _ 0 UP UNTIL > last DO                                    01531
         BEGIN                                                        01532
         %move one entry%                                             01533
            j _ IF j <= 0 THEN last ELSE j -1;                        01534
            &sre _ base + (srrelen*j);                                01535
```

```
            IF j = top THEN EXIT LOOP;                                  01536
          %continue backing up until find valid entry%                  01537
            count _ 0;                                                  01538
            UNTIL sre.srexis DO                                         01539
               BEGIN                                                    01540
               j _ IF j <= 0 THEN last ELSE j -1;                       01541
               &sre _ base + (srrelen*j);                               01542
               IF j = top THEN EXIT LOOP 2;                             01543
               IF (count _ count+1) > last THEN                         01544
                  RETURN(0);                                            01545
               END;                                                     01546
         END;                                                           01547
      RETURN(i);                                                        01548
      END.                                                              01549
                                                                        01550
%.....Execute Edit Routines.......%                                     01551
   (editx) PROCEDURE (ptr);                                             01552
      %-----------------%                                               01553
      LOCAL                                                             01554
         char, %last character read%                                    01555
         numb,                                                          01556
         insrtf;                                                        01557
      REF ptr;                                                          01558
      LOCAL TEXT POINTER z1, z2, ptr2;                                  01559
      CCPOS SF(ptr);                                                    01560
      *lit* _ NULL; %INITIALISE STRING%                                 01561
      crlf();                                                           01562
      echoff();                                                         01563
      insrtf _ FALSE;                                                   01564
      %deactivate ^O and ^S interrupts%                                 02653
         r1 _ 4B5; r2 _ 240000004000B; !JSYS dic;                       02654
         r1 _ 15;  !JSYS dti;  %deassign terminal code for ^O%          02655
         r1 _ 16;  !JSYS dti;  %deassign terminal code for ^P%          02656
         r1 _ 17;  !JSYS dti;  %deassign terminal code for ^Q%          02657
         r1 _ 19;  !JSYS dti;  %deassign terminal code for ^S%          02658
      LOOP                                                              01569
         CASE char _ input() OF                                         01570
            =BC %^H%, =$ctla: %backspace character%                     01571
               BEGIN                                                    01572
               bkc($lit);                                               01573
               todco(BC);                                               01574
               END;                                                     01575
            =CA %^D%, =C. %^B%: %copy remainder of old to new and
            terminate alter%                                            01576
               BEGIN                                                    01577
               todco(char);                                             01578
               copych(2000, typefg);                                    01579
               EXIT;                                                    01580
               END;                                                     01581
            =$ctle: %change source of text to or from keyboard and type
            < or >%                                                     01582
               IF insrtf THEN                                           01583
                  BEGIN                                                 01584
                  insrtf _ FALSE;                                       01585
                  typech(">);                                           01586
                  END                                                   01587
```

```
   ELSE                                                          01588
      BEGIN                                                      01589
      insrtf _ TRUE;                                             01590
      typech("<");                                               01591
      END;                                                       01592
=$ctlf: %copy 1 chr and type%                                    01593
    IF copych(1) THEN EXIT;                                      01594
=$ctlg: %skip up thru c and type percent ptr%                    01595
   IF (numb _ search(input())) THEN skip(numb)                   01596
   ELSE typeas($" ?");                                           01597
=$ctln: %backspace character in old and new%                     01598
   BEGIN                                                         01599
   bkc($lit);                                                    01600
   typech("^");                                                  01601
   swork1 _ MAX(1, swork1-1);                                    01602
   fechc1(forward, $swork);                                      01603
   END;                                                          01604
=$ctlo: %copy to c%                                              01605
   IF (numb _ search(input())) THEN copych(numb-1)               01606
   ELSE typeas($" ?");                                           01607
=$ctlp: %skip up to c and type percent sign %                    01608
   IF (numb _ search(input())) THEN skip(numb-1)                 01609
   ELSE typeas($" ?");                                           01610
=$ascbst %^Q%: %backspace statement in old and new%              01611
   BEGIN                                                         01612
   todco($ascbst);                                               01613
   crlf();                                                       01614
   *lit* _ NULL;                                                 01615
   CCPOS SF(ptr);                                                01616
   END;                                                          01617
=$ctlr: %retype line up to this point%                           01618
   BEGIN                                                         01619
   crlf();                                                       01620
   typeas($lit);                                                 01621
   END;                                                          01622
=$ctls: %skip one character on old statement--type percent%
                                                                 01623
    IF skip(1) THEN EXIT; %end of statement%                     01624
=$ctlu: %copy thru end of old statement%                         01625
   copych(2000);                                                 01626
=BW, =$ctlw: %backspace word in new%                             01627
   BEGIN                                                         01628
   <NLS, UTILTY, bkw>($lit);                                     01629
   todco(BW);                                                    01630
   END;                                                          01631
=CD %^X%:                                                        01632
   BEGIN                                                         01633
   edixqt();                                                     01634
   GOTO STATE;                                                   01635
   END;                                                          01636
=$ctly: %repeat alter%                                           01637
   BEGIN                                                         01638
   copych(2000, typefg);                                         01639
   FIND SF(*lit*) ^z1 SE(*lit*) ^z2;                             01640
   FIND SF(ptr) ^ptr SE(ptr) ^ptr2;                              01641
   creptex($ptr, $ptr2, $z1, $z2);                               01642
```

```
            tda.dacsp _ ptr;                                        01643
            tda.daccnt _ 1;                                         01644
            REPEAT($ascbst);                                        01645
            END;                                                    01646
         =$ctlz: %copy thru c%                                      01647
            IF (numb _ search(input())) THEN copych(numb)           01648
            ELSE typeas($" ?");                                     01649
         =$ascalt: % terminate alter%                               01650
            EXIT;                                                   01651
         ENDCASE %other character..insrtf into new%                 01652
            BEGIN                                                   01653
            todco(char);                                            01654
            *lit* _ *lit*, char;                                    01655
            IF NOT insrtf THEN char _ READC;                        01656
            END;                                                    01657
   edixqt();                                                        01658
   RETURN;                                                          01659
   END.

                                                                    01660
(edixqt) PROCEDURE;                                                 02659
   %reassign the terminal codes%                                    02660
      r1 _ 230000018; %^S%                                          02661
      !JSYS ati;                                                    02662
      r1 _ 210000308; %^Q%                                          02663
      !JSYS ati;                                                    02664
      r1 _ 200000018; %^P%                                          02665
      !JSYS ati;                                                    02666
      r1 _ 170000038; %^O%                                          02667
      !JSYS ati;                                                    02668
   r1 _ 485;                                                        02669
   r2 _ 2400000040008; %activate ^S, ^O, ^P, ^Q%                    02670
   !JSYS aic;                                                       02671
   RETURN END.

                                                                    02672
(copych) PROCEDURE (numb);                                          01671
   %copy 'numb' characters from old to new%                         01672
   %----------------%                                              01673
   LOCAL char;                                                      01674
   UNTIL (numb := numb - 1) <= 0 DO                                 01675
      CASE char _ READC OF                                          01676
         =ENDCHR: RETURN(TRUE);                                     01677
         ENDCASE                                                    01678
            BEGIN                                                   01679
            *lit* _ *lit*, char;                                    01680
            typech(char);                                           01681
            END;                                                    01682
   RETURN(FALSE);                                                   01683
   END.

                                                                    01684
(search) PROCEDURE (target);                                        01685
   %search old statement for character in target.  return numb of
   characters away from current swork setting if found, 0 if not
   found%                                                           01686
   %----------------%                                              01687
   LOCAL count, save, last, char;                                   01688
   count _ 1;                                                       01689
```

```
      save _ swork[1];                                                  01690
      LOOP                                                              01691
         CASE READC OF                                                  01692
            = ENDCHR:                                                   02677
               BEGIN                                                    02678
               swork[1] _ save;                                         02679
               fechc1(forward, $swork);                                 02680
               RETURN(FALSE);                                           02681
               END;                                                     02682
            = target:                                                   01694
               BEGIN                                                    01695
               swork[1] _ save;                                         01696
               fechc1(forward, $swork);                                 01697
               RETURN(count);                                           01698
               END;                                                     01699
            ENDCASE BUMP count;                                         01700
      END.
                                                                        01701
   (skip) PROCEDURE (numb);                                             01702
      %skip a numb of characters in old and type percent signs%         01703
      %-----------------%                                               01704
      LOCAL last, char;                                                 01705
      UNTIL (numb _ numb - 1) < 0 DO                                    01706
         CASE READC OF                                                  01707
            =ENDCHR: RETURN(TRUE);                                      01708
            ENDCASE typech("%");                                        01709
      RETURN(FALSE);                                                    01710
      END.
                                                                        01711
%.....STARTUP ROUTINES....%                                             01713
   (dstart)     % DNLS startup routine %                                01714
      PROCEDURE(gs);                                                    01715
      IF nldevice = devlproc THEN                                       01754
         BEGIN                                                          01755
         dsubsys($" ");                                                 01716
         pad(lpxmax);                                                   01758
         dn($" ");                                                      01717
         pad(lpxmax);                                                   01756
         END;                                                           01757
      dpset(dspallf,endfil,endfil,endfil); %set display parameters%
                                                                        01718
      recred();%total recreate display%                                 01719
      IF gs THEN dismes(5000,gs); %put up greetings for 5 seconds%      01720
      supervisor();                                                     01721
      halt();                                                           01722
      END.
                                                                        01723
   (tstart)     % TNLS startup routine %                                01734
      PROCEDURE(gs);                                                    01735
      %-----------------%                                               01736
      LOCAL stid;                                                       01737
      IF gs THEN %there is a message to print%                          01738
         BEGIN                                                          01739
         crlf();                                                        01740
         typeas(gs);                                                    01741
         crlf();                                                        01742
```

```
    END;                                                              01743
IF exquery THEN quin(nlparse) ELSE supervisor(); %start command
parsing%                                                              01744
halt();   % terminate and prepare for re-entry %                      01745
END.                                                                  01746
                                                                      01747
FINISH of frontend                                                    01748
```

GRUNLDR.

```
< NLS, GRUNLDR.NLS.16, >, 23-Apr-78 20:17 JDH ;;;;
;Output Assembler File (rel-nls,grunldr.txt,)                    083
CONN REL-NLS                                                    084
DA<ESC>                                                         085
<NETSYS>TENLDR                                                  086
/S                                                              087
/3777775                                                        088
/W                                                             0163
/M                                                              089
/117E                                                           090
/4000100                                                        091
PDATA                                                           092
SYNTAX                                                          094
PARSER                                                          095
SELECT                                                          096
PSUPPORT                                                        097
PSEDIT                                                          098
PSSENDMAIL                                                      099
PSPROGS                                                        0100
PSUSEROP                                                       0102
PSSYNGEN                                                       0104
FONLY                                                          0105
PSSYSTEM                                                       0101
PSHELP                                                         0103
FINTNLS                                                        0106
TSPRT                                                          0107
PRMSPC                                                         0108
PPOVER                                                         0109
/5520000                                                       0110
UPDATA                                                         0111
/1400                                                          0112
BDATA                                                          0113
FDATA                                                          0114
SDATA                                                          0115
L10DATA                                                        0116
NDDTDATA                                                        093
UNDATA                                                         0117
FCONST                                                         0118
BCONST                                                         0119
FRECORDS                                                       0120
BRECORDS                                                       0121
SRECORDS                                                       0122
L10RUNTIME                                                     0123
UTILTY                                                         0124
BINTNLS                                                        0125
SINTNLS                                                        0126
CEDIT1                                                         0127
CEDIT2                                                         0128
FILMNP                                                         0129
SEQGEN                                                         0130
FRONTEND                                                       0133
IOEXEC                                                         0135
ADRMNP                                                         0134
CORESUPPORT                                                    0136
CALCSUPPORT                                                    0137
VERIFY                                                         0138
```

```
SEQFIL                                                          0139
AUXCOD                                                          0140
OPTAB                                                           0141
NDDT2                                                           0142
COLSRT                                                          0143
RECFIL                                                          0144
IDENTSUPPORT                                                    0145
CSENDMAIL                                                       0146
CATNUM                                                          0151
DSPGEN                                                          0131
CSYNGEN                                                         0149
EXECFL                                                          0132
CPROGRAMS                                                       0147
CHELP                                                           0148
INPFBK                                                          0150
STGMGT                                                          0152
SYNTBL                                                          0153
STENEX                                                          0154
/120R                                                           0155
<ESC>                                                           0156
MERGE <NETSYS>ARCDDT.SAV                                        0157
DDT                                                             0158
MOVE 1,116<ESC>XMOVEM 1,<ESC>I-1<ESC>X                          0159
BRKHER<ESC>8B HALTF<ESC>X                                       0160
CSAVE <REL-NLS>GNLS.EXE.<ESC>0<ESC>777777                       0164
                                                                0161
POP                                                             0162
```