1 LUNDI.

FCONST FCONST FCONST FCONST FCONST FCONST FCONST FCONST
FCONST FCONST FCONST FCONST FCONST FCONST FCONST FCONST

```
< NLS, FCONST.NLS;6, >, 1-OCT-74 17:42 KEV ;;;;
FILE fconst    % L10  <REL-NLS>fconst %% (L10,) (rel-nls,fconst.rel,) %
                                                                        0225
DECLARE EXTERNAL STRING                                                 0223
   intca = "Type CA to proceed";                                        0224
DECLARE EXTERNAL                                                         06
   cmdelete = 4934768,                                                   07
   popstate=12345,                                                       08
   interperr=3614B, % cfd's extension %                                  09
   gaderr  = 427,          % TNLS address error %                        010
   statesig = 7453342B,    % For simulated GOTO STATE's%                 011
   errsig= 0 ;                                                          012
DECLARE EXTERNAL  % case shift 2 mouse-keyset translation table %       0226
   cssh2 = (                                                            0227
      '!, '", '#, '$, '%,                                               0228
      '&, '', '(, '), '@,                                               0229
      '+, '-, '*, '/, '↑,                                               0230
      '0, '1, '2, '3, '4,                                               0231
      '5, '6, '7, '8, '9,                                               0232
      '=                                                                0233
   );                                                                   0234
DECLARE EXTERNAL  % mouse button translation table %                    0235
   msetbl = (                                                           0236
      0,                                                                0237
      CA,                                                               0238
      CD,                                                               0239
      C.,                                                               0240
      BC,                                                               0241
      ALT,                                                              0242
      BW,                                                               0243
      0                                                                 0244
   );                                                                   0245
SET EXTERNAL %character codes and group names%                           013
   %character codes for special characters%                             014
      ascbst =   21B, % backspace stmt %                                015
      asctsw =  36B,  % text switch (used by todas execute text) %      016
      ctla   =  1,    % control a %                                     017
      ctle   =  5,    % control e %                                     018
      ctlf   =  6,    % control f %                                     019
      ctlg   =  7,    % control g %                                     020
      ctln   =  16B,  % control n %                                     021
      ctlo   =  17B,  % control o %                                     022
      ctlp   =  20B,  % control p %                                     023
      ctlr   =  22B,  % control r %                                     024
      ctls   =  23B,  % control s %                                     025
      ctlt   =  24B,  % control t %                                     026
      ctlu   =  25B,  % control u %                                     027
      ctlv   =  26B,  % control v %                                     028
      ctlw   =  27B,  % control w %                                     029
      ctlx   =  30B,  % control x %                                     030
      ctly   =  31B,  % control y %                                     031
      ctlz   =  32B,  % control z %                                     032
      ascalt =  33B;  % altmode %                                      033
% TENEX things %                                                        039
   DECLARE EXTERNAL                                                     040
   % file access bit definitions %                                      041
```

```
    racc=100000B6,    % read access(bit 2) %                          042
    wacc=40000B6,     % write access(bit 3) %                         043
    eacc=20000B6,     % execute access(bit 4) %                       044
    tuacc=1000B6,     % trap to user on access(bit 8) %               045
    cwacc=400B6;      % copy on write access(bit 9) %                 046
% File Things %                                                       047
  DECLARE EXTERNAL STRING                                            051
    nlsext = "NLS",                                                  052
    savext = "SAV",                                                  05
    ctlext = "CTL";                                                  054
  DECLARE EXTERNAL                                                   055
    gtjoof=4B11, gtjoif=1B11, gtjprf=1B11, gtjoosf=4B11, gtjoisf=1B11,
    gtjstr = 1B8, gtjprv = 2B9, gtjidl = 1B9, gtjnfo = 2B11,         056
    write = 0, read = 1, readwrite = 2, append = 3, rthawed =
    5,rwthawed = 4;                                                  057
% parser %                                                          058
  % EXTERNL KEYWORDS (used as types for the SELECTION processor) %   059
    DECLARE EXTERNAL STRING % so only one copy exists in system These
    keywords are defined as external strings in CONST. %             060
      % STRUCTURAL ENTITIES %                                        061
        branch = "BRANCH",                                           062
        group = "GROUP",                                             063
        plex = "PLEX",                                               064
        statement = "STATEMENT",                                     065
      % TEXTUAL ENTITIES %                                          066
        character = "CHARACTER",                                     067
        invisible = "INVISIBLE",                                     068
        link = "LINK",                                               069
        number = "NUMBER",                                           070
        password = "PASSWORD",                                       071
        text = "TEXT",                                               072
        visible = "VISIBLE",                                         073
        word = "WORD",                                               074
      % MISC. ENTITIES %                                            075
        file = "FILE",                                               076
        oldfilelink = "OLDFILELINK",                                 077
        newfilelink = "NEWFILELINK",                                 078
        name = "NAME",                                               079
        return = "RETURN",                                           080
        filereturn = "FILERETURN",                                   081
        ident = "IDENT",                                            0247
        identlist = "IDENTLIST",                                    0246
        edge = "EDGE",                                               082
        marker = "MARKER",                                           083
        rest="REST", menu="MENU", hlpcom="HLPCOM", cntlq="CNTLQ",
        back="BACK"; % for HELP %                                   0249
  % SUBSYSTEM PROCESSING MODES %                                     092
    DECLARE EXTERNAL                                                 093
      sbstart = 1,  % subsystem initialization %                     094
      sbrun = 2,    % command execution of subsystem %              095
      sbfinish = 3,         % termination of a subsystem %          096
      sbpop = 4,    % backup after termination %                    097
      sbrentry = 5;         % rentry after leaving a subsystem %     098
  % COMMAND RECOGNITION MODES %                                      099
    DECLARE EXTERNAL                                                0100
      mexpert = vexpert, % default, NLS style %                     0101
```

```
      mfixed = 5,    % fixed number of chars %                          0102
         fixl = 3,  % number of chars required for fixd mode %          0103
      mdemand = 2,           % TELNET model with right delimiters %
                                                                        0104
      manticipitory = 3;     % minimum unique, no right delimiters %
                                                                        0105
% COMMAND HERALD MODES %                                               0106
   DECLARE EXTERNAL                                                     0107
      onechar = 1,           % only single character hearald %         0108
      multchar = vmultchar;          % multiple character hearald      0109
         (length is hrldsize) %                                         0110
% INTERPRETER PARSING MODES %                                          0111
   DECLARE EXTERNAL                                                     0112
      parsing = 1,           % normal parsing mode %                   0113
      pnext = 2,             % get successor mode %                    0114
      cleanup = 3,           % termination of a command %              0115
      rptparse = 4,          % repeat of a command %                   0116
      backup = 5,            % backup after parse fail %               0117
      popselect = 6,         % pop one selection %                     0118
      parsehelp = 7,         % solicit help info %                     0119
      parseqmark = 8;        % solicit questionmark string %           0120
% PROMPT MODES (VALUES FOR GLOBAL INPROMPTS) %                         0121
   DECLARE EXTERNAL                                                     0122
      noprompts=1,           % prompts off %                           0123
      partprompts=2,         % no optional prompting %                 0124
      fullprompts=vfullprompts;      % all prompts %                   0125
% FEEDBACK MODES %                                                     0126
   DECLARE EXTERNAL                                                     0127
      verbsmode=vverbsmode,                % give em everything %
                                                                        0128
      tersemode=1;           % no noise word feedback, dont fill out
      keywords %                                                        0129
% FUNCTION CODES FOR FBCTL %                                           0130
   DECLARE EXTERNAL                                                     0131
      clearcfl = 1001,       % clear cfl %                             0132
      addchar = 1002,        % add char to cfl %                       0133
      % keyword = 1003,       add keyword to cfl %                    0134
      startrec = 1004,       % start keyword recognition %             0135
      echostr = 1005,        % echo a given string in the feedback
      area %                                                            0136
      rechostr = 1006,       % replace a given string in the feedback
      area %                                                            0137
      typelit = 1007,        % type out supplied string %              0138
      startparams = 1008,    % begin parameter collection %            0139
      incues = 1009,         % provide inpt prompts %                  0140
      fbpop = 1010,                % cleanup after popping states %
                                                                        0141
      typecalit = 1011,      % wait for CA after printing literal %
                                                                        0142
      fbaddlit = 1012,       % add string to literal feedback area %
                                                                        0143
      addcalit = 1013,       % wait for CA %                           0144
      typenulllit = 1014,    % type null literal string %              0145
      fbendlit = 1015;       % type lit and wait for input %           0146
% RECORD SIZE DECLARATIONS %                                           0147
   DECLAE EXTERNAL                                                     0148
```

```
      d2sel = 2,     % displacement of second selection %                0149
      pselLen = 4;  % length of pselrecord %                            0152
% user-options defaults %                                               0160
   DECLARE EXTERNAL                                                     0161
   % view specs %                                                       0162
      defvs1,                % default viewspecs - word 1 %             0163
      defvs2;                % default viewspecs - word 2 %             0164
% Device things %                                                       0165
   DECLARE EXTERNAL                                                     0166
   % devices and device modes %                                         0167
      %device types%                                                    0168
         ctty = 100B,                    %controling tty number%       0169
         arcoprtty = 27B,    %arc operator's tty number%               0170
         utiloprtty = 1B,    %utility operator's tty number%           0171
         dev33 = 0,                                                     0172
         dev35 = 1,                                                     0173
         dev37 = 2,                                                     0174
         devtiex = 3,                                                   0175
         devsri = 4,                     %tasker displays%              0176
         imlac0 = 5,                                                    0177
         imlac1 = 6,                                                    0178
         netTty=7,                                                      0179
         offline=101,                                                   0180
         dev1proc = 11,      %value of nldevice for line processors%
                                                                        0181
      %modes%                                                           0182
         typewriter=0, fulldisplay=1;                                   0183
   %imlac protocal%                                                     0184
      DECLARE EXTERNAL remfudge = 40B;                                  0185
      DECLARE EXTERNAL %control codes for remote displays (e.g.
      imlacs)%                                                          0186
         begmsg = 33B, %begin message character%                        0187
         echda = 3, %alternate seq da echoing%                          0188
         remsdda = 6, %suppress display area control character%         0189
         remrdda = 7, %restore display area control character%          0190
         remrsda = 11B, %restore string in display area control
         character%                                                     0191
         remtsn = 12B, %tty simulation on control character%            0192
         remtsf = 13B, %tty simulation off control character%           0193
         remlcm = 14B; %switch to long character mode%                  0194
      DECLARE EXTERNAL rinistr =(154663315466B, 154663315466B,
      404B9);                                                           0195
         %string of 10 reminits plus 1 char%                           0196
   % Display stuff %                                                    0197
      DECLARE EXTERNAL                                                  0198
         syserr = 0, rtmax = 60;                                        0199
      DECLARE EXTERNAL                                                  0200
         current=-1;                     %use current x,y position%     0201

                                                                        0202
   % jump return ring sizes%                                           0203
      DECLARE EXTERNAL                                                  0204
         frrmax=25, %max entries in file return ring%                   0205
         srrmax=25; %max entries in statement return ring%              0206
   % Use measurement%                                                  0213
      DECLARE EXTERNAL                                                  0214
```

```
        dntyp=0, %indicates user running display NLS%                    0215
        tntyp=1, %indicates user running typewriter NLS%                 0216
        nttyp=2,                                                         0217
        dspallf = 6;%reformat every stmt, all da's% %indicates user
        running network NLS%                                             0218
      DECLARE EXTERNAL                                                   0219
        orgstid = 2; %STID of origin with stfile=0%                      0220
% HELP constants %                                                       0248
  DECLARE EXTERNAL stkmax = 26,                                          0250
    entind = -1,                                                         0251
    rngmax = 26,                                                         0252
    rngint = 2;                                                          0253
FINISH                                                                   0221
```

ΓΥΑΙΑ.

(MLK)FDATA (MLK)FDATA (MLK)FDATA (MLK)FDATA (MLK)FDATA (MLK)FD

```
< NLS, FDATA.NLS;19, >, 11-OCT-74 16:44 DSM ;;;;
FILE fdata   %  LIO   <REL-NLS>FDATA   %% (LIO,) (rel-nls,FDATA.rel,) %
                                                                    0353
DECLARE EXTERNAL                                                    0348
   xacs [16],  % ACs at entry to NLS %                              0349
   intmsf=0;  %flag for message at initialization time%             0350
DECLARE EXTERNAL STRING                                             0351
   intmsg[500]; %String for entry message%                          0352
REGISTER %here so DDT will use these on printout%                    03
   r1=1, r2=2, r3=3, r4=4, r5=5, p=7, wa=8, s=9, m=10, rp=11,        04
   a1=12, a2=13, a3=14, a4=15:                                       05
% stack sizes %                                                     06
   SET EXTERNAL                                                     08
      %call and multiple result stack lengths%                      09
         gstksz =  3072,  %length of general call stack%           010
         pstksz =  20,    %length of pattern stack%                011
      %private stack for illegal intruction pseudo-interrupt%      012
         psisksz = 100;                                            013
% *** GLOBAL DATA FOR PAGE 0 *** %                                 014
% Special stuff declared in page 0, loaded at 140B %               015
   %...KEEP THESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%     016
   DECLARE EXTERNAL %...important stuff...%                         017
   %...user related data...%                                        018
      cuno,    % current user number %                             020
      nldevice, %current terminal device%                          021
      console, % octal console number %                            022
   %...set command and aptstr routine...%                           023
      xsmode=#0B,     %  case set    %                            025
      modeon=0;       %  option mask %                            027
   % save stack for viewspec input backup%                         028
      DECLARE EXTERNAL STACK savevspec[20, 2];                      029
DECLARE EXTERNAL                                                    031
   nlcron=1; %measurement with nlscr jsys on/off%                   032
   % stack for illegal instruction pseudo-interrupt %              034
      DECLARE EXTERNAL psistk[100];                                 035
   %Bit tables for LSID and DAID assignment%                        036
      DECLARE EXTERNAL                                              037
         dabt[3],     %daid bit table%                              038
         lsbitables[56] %7 wrds, 8 das%,  %lsid bit tables for file
         text display areas, pointed to by dalsidb field in da records%
                                                                    039
         clsbitables[56] %7 wrds, 8 das%,  %lsid bit tables for control
         info areas, pointed to by dalsidb field in da records%     040
         lsbtsize = 7,  %size (number of words) of a lsbitable%     041
         dabtsize = 3;  %size (number of words) of the dabt%        042
   % useroptions %                                                  043
      DECLARE EXTERNAL                                              044
         uojfn;          % jfn for the user profile file %          045
% DECLARE's %                                                       049
   DECLARE EXTERNAL                                                 050
      linkcnsl=0,     %line number of console when display linked%  052
      savnldevice,   %used to save nldevice during screen sharing%  053
      savedda,  %address of DNSL display area "Taken" to TNLS, CED% 054
      savetda,  %address of TNSL display area "Taken" to DNLS, CED% 055
      ttysim=0,   %daid of ttysimulation window in DNLS%            056
      ttyda,     %address od da currently being used as tty da%     057
```

```
   defttysim=1, %FLAG:                                                  058
      TRUE: default tty-sim area being used,                            059
      FALSE: user-supplied tty-sim area being used%                     060
   nlssbn,    %contains six-bit name for version of nls being used%
                                                                        063
   jnlsbn=0,  %contains six-bit name for journal%                       064
%...display area stuff.....%                                           065
   dpyarea[160], %dal*damax%                                            066
      dpyend,                                                           067
   nlsdas[160], %for cfl, name area, lit area, msg area, date-time
   area, view spec areas, etc%                                          068
      %CAN REDUCE THIS BY TWO*DAL WHEN MSGDA AND LITDA GO AWAY%         069
      nlsdae,                                                           070
   dacnt = 0,    %0 => no da's active%                                  071
   rtfree = 0,   %free list pointer--physical entry count%              072
   vspsav[2],    %save previous viewspecs%                              073
   bmcnt=0,        %number of bugmarks on the screen%                   074
%...pbug support.....%                                                 075
   pbugtb[40],                                                          076
   pbugte,                                                              077
   pbugpt,                                                              078
%...system handles for display areas...%                               083
   tda,      %TNLS display area address%                                084
   tada,     %text area display area record address%                    085
   cflda,    %Command Feedback Line display area record address%        086
   echoda,   %echo register (not used) display area record address%
                                                                        087
   ltvsda,   %L-T viewspec display area record address%                 088
   vspcda,   %"hjuCP" view spec display area record address%            089
   namda,    %name area display area record address%                    090
   subda,    %subsystem name display area record address%               091
   litda,    %literal feedback display area record address%             092
   msgda;    %message display area record address%                      093
DECLARE EXTERNAL STRING                                                094
   rmdcr   =  "↑↑↑↑",    %initialize as up arrow%                      095
   drmdcr  =  "++++",     %initialize as plus%                          096
   spacestr   =  "
                  ",                                                    097
   altdir[30],    %alternate directories for rdlit%                    0366
   altext[50],    %alternate extensions for rdlit%                     0367
   cflstr[100],    %Command Feedback Line%                              098
   savcfl[100],     %previous Command Feedback Line%                    099
   cflarw[25],      %arrow and question mark under CFL%                 0100
   vspstr[30],  %viewspec string%                                      0101
   initsr[15],  %user's ident%                                         0102
   nlsnam[10], %contains name of nls system being used%                0103
   cdstr1[2000],  %used by create display for intra-statement edits%
                                                                       0104
   cdstr2[2000],  %used by create display for intra-statement edits%
                                                                       0105
   conreg[100],  %jump string save area%                               0106
   wrdreg[40],  %jump string save area%                                0107
   namreg[40],  %name default register%                                0108
   littabs[100]; %used in DNLS literal collection for tabs%            0109
DECLARE EXTERNAL                                                       0110
   sysbreak[128],   % break character transfer vector for rdlit %
```

```
                                                                    0111
   trnslo[128],      % typewriter output translation table %       0112
   trnsli[128];      % typewriter input translation table %        0113
DECLARE EXTERNAL                                                    0114
   altdfl, %flag indicating rdlit should use alternate directories %
                                                                    0368
   litdahandle,                                                     0115
   litline,     %index into column%                                 0116
   lplitline,     %saved value of litline for line processor
   terminals%                                                       0421
   lplitreset =0,     %reset literal flag for line processor%       0420
   litrmarg,    %column number of lit area rightmargin%             0117
   spacecol,    %column which had last SP (or non-printing char) for
   DNLS literal collection%                                         0118
   litstore,    %byte pointer for rapid string appends and line
   breaks%                                                          0119
   litstring,  %address of string being displayed in literal area%
                                                                    0120
   litcolumn[35],  %column array for DNLS literal collection%       0121
   tabstore;  %byte pointer into string TABS%                       0122
DECLARE EXTERNAL %...buff...%                                       0123
   buff[40],    %character input buffer -- array for speed%         0124
   buffs  = 0,       %strating buffer index%                        0125
   buffn = 0,        %next buffer index%                            0126
   buffct,      %index into buff and bufffg%                        0127
   buffsz = 39; %max index into buff%                               0128
DECLARE EXTERNAL STACK                                              0131
   bugmarks[20], fvstk[65 %svmxlev+1%];                             0132
%..... HELP Declarations..... %                                     0370
   DECLARE EXTERNAL helploc[4]; % fake subsystem dispatch record %
                                                                    0371
   DECLARE EXTERNAL calcaux;                                        0373
   DECLARE EXTERNAL inhelp=0; % TRUE if in help command, FALSE
   otherwise. %                                                     0374
   DECLARE EXTERNAL qagain=0; % TRUE if in cntl-q typed while in
   help command, FALSE otherwise. %                                 0375
   DECLARE EXTERNAL hlpfileno;                                      0376
   DECLARE EXTERNAL qda, qsw, qstorblk, qnewstmt;                   0377
   DECLARE EXTERNAL qdavspc, qdavs2;                                0378
   DECLARE EXTERNAL                                                 0379
      overscreen, % TRUE if screen overflows. %                     0380
      hseqer = 0; % TRUE if trouble in help sequence generator %    0381
   DECLARE EXTERNAL multiflg=0, sflg=1 %search type%;               0382
   DECLARE EXTERNAL                                                 0383
      menchr = ']', comchr = '%', % Format characters %             0384
      mentyp=1, multyp=2, untyp=0; % Format type of statement %
                                                                    0385
   DECLARE EXTERNAL                                                 0386
      moremen, % TRUE if more menu to be shown. %                   0387
      conrng, % The address of the context ring block. %            0388
      curstk, curindex, % The address of the current context ring
      element %                                                     0389
      confre; % The next free context index. %                      0390
   DECLARE EXTERNAL newstk = 1; % flag for qmenu.  calls pushent for
   new context stack only if TRUE %                                 0391
   DECLARE EXTERNAL topcon[2], entcon[2]; % Contexts for entry and
```

```
    top. %                                                              0392
    DECLARE EXTERNAL                                                    0393
       qspecs,     % current query viewspecs in force %                 0394
       qpspecs;     % previous query viewspecs in force %               0395
    DECLARE EXTERNAL                                                    0399
       qmenumax = 15,                                                   0400
       qmenucnt = 0,    % current value for NEXT menu item %            0401
       qcclwidth = 24;                                                  0402
    DECLARE EXTERNAL                                                    0396
       hdebug = 1; %true for debugging...loads xhelp%                   0397
  %...frozen statement list...%                                         0139
    DECLARE EXTERNAL                                                    0398
    fzlsts[80],                                                         0140
       fzlste, %end of buffer%                                          0141
    fzfree, %free list pointer for fzlsts%                              0142
    frzmax = 20; %length of freeze list%                                0143
  %...input stuff...%                                                   0157
    DECLARE EXTERNAL STRING altinp[100];  %LP startup alternate
    input%                                                              0361
    DECLARE EXTERNAL                                                    0362
    msdbit = 0,        % mouse button dirty bit %                      0355
    curmbt = 0,        % current state of mouse buttons %              0356
    ordmbt = 0,        % or'd state of mouse buttons %                 0357
    gcords,            % coordinates %                                 0354
    bugreg[2],        %bug mark area%                                   0158
    cflpos,           %command feedback column counter%                 0159
    csrarmd=1,          %cursor armed-disarmed flag%                    0160
    arowon=0,           ?/CFL arow on-off flag%                          0161
    ltsmall=1,          %lt-viewspec area large-small flag%             0162
    qmrkon=0,           %CFL question mark on-off flag%                 0163
    namenull=1,         %TRUE if name area is null (empty)%             0164
    litreset=1,         %TRUE if literal area is reset%                 0165
    littyflag=0,        %TRUE if lit area is being used as a tty
    window%                                                             0166
    litapflag=0,        %TRUE if lit area in append mode%               0167
    msgreset=1,         %TRUE if message area is reset%                 0168
    namereset=1,    %TRUE if name area is reset%                        0169
  %...control environment stuff...%                                     0177
    inpwatchjfn = 0, %jfn for output control file%                      0178
    inpjfn = 0, %jfn for control file input; 0 means use controlling
    tty%                                                                0179
  %...work station char input buffer...%                                0180
    curchr,     %curent char--after  markers have been resolved%        0181
    rawchr,     %address of lowest level character input routine%       0182
  %...auxilary input stuff...%                                          0183
    auxsav,     % saved "rawchr" dispatch %                            0184
    auxmod,     % saved recogmode when reading aux. input %            0185
    auxmd2,     % saved reco2mode when reading aux. input %            0186
    auxtp1[2], % text ptr to start of auxilary text %                   0187
    auxtp2[2], % text ptr to end of auxilary text %                     0188
    auxwrk[7], %executable text work area%                              0189
%...global display/print parameters...%                                 0190
  %...view parameters and values...%                                    0191
    %...View specs...%                                                  0192
       sysvspec[2],    %view spec flags and values%                     0193
    lvadjnm,   %if true then print statement numbers during levadj %
```

```
                                                                    0194
%...parameters to dpset for selective recreate display%             0199
   cdstd1,  %stid of stmt to be reformatted, if cdtype = dsprfmt  or
   dsprfst; else passed as file indicators so da's may be
   selectively updated%                                             0200
   cdstd2,  %second stid or file indicator%                         0201
   cdstop,  %after encountering this stid, daupdate knows no
   further structural or editing changes took place on the old lsrt
   entries.%                                                        0202
   cdtype:  %option to inform daupdate of general nature of
   operation performaed by command issued.  Options are defined in
   data.%                                                           0203
DECLARE EXTERNAL                                                    0204
%...general global variables...%                                    0205
   continue=0, %  continue flag--set by terminate  %                0206
   nlmode,     %= typewriter or NOT typewriter%                     0207
   nlparse=1, %TRUE for non-query entry; FALSE for query%           0208
   exquery,                  % no ident or initial file needed %    0209
   typefg = 1,     %type flag for alter%                            0210
   cmarkflg = 0,   %show control marker flag%                       0211
   echofg,         %typewriter echo flag for output%                0212
   ctrlchar,                 % if TRUE echo control chars as <↑char>
   %                                                                0213
   oshift,         %typewriter output shift mode%                   0214
   cshift,         %character which causes lower to upper case shift
   in TNLS for the following character -- see <inpfbk, tinptc>%     0215
   wshift,         %character which causes lower to upper case shift
   in TNLS for the following word -- see <inpfbk, tinptc>%          0216
   wordshift,      %Flag used by <inpfbk, tinptc> for doing word
   shifts%                                                          0217
   todlit,         %typewriter literal escape character%            0218
%...initial values for display areas...%                            0219
   rmarg, %right margin%                                            0220
   lmarg, %left margin%                                             0221
   tmarg, %top margin%                                              0222
   bmarg, %bottom margin%                                           0223
   %text area%                                                      0224
      tabase = taleft,                                              0225
      taleft = 0, %lmarg%                                           0226
      taright = 0, %rmarg%                                          0227
      tatop = 0, %tmarg + 9*tavinc%                                 0228
      tabottom = 0, %bmarg%                                         0229
      tacsize = 1, %character size%                                 0230
      % tafont = 0, font is normal%                                 0231
      tahinc = 0, %14*256%                                          0232
      tavinc = 0, %30*256%                                          0233
      tax = 0,                                                      0234
      tay = 0,                                                      0235
      tamind = 15,                                                  0369
   %Command Feedback Line (x=300B, y=1672B)%                        0236
      cflbase = cflleft,                                            0237
      cflleft = 0, %lmarg + 20*cflhinc%                             0238
      cflright = 0, %cflleft + 40*cflhinc%                          0239
      cfltop = 0, %tmarg + 3*cflvinc%                               0240
      cflbottom = 0, %cfltop + 2*cflvinc%                           0241
      cflcsize = 1, %character size%                                0242
```

```
    cflfont = 0, %font is normal%                                   0243
    cflhinc = 0, %14*256%                                           0244
    cflvinc = 0, %30*256%                                           0245
    cflx = 0,                                                       0246
    cfly = 0,                                                       0247
%Name area (x=600, y=1672B)%                                        0248
    nambase = namleft,                                              0249
    namleft = 0, %cflright + cflhinc%                               0250
    namright = 0, %rmarg%                                           0251
    namtop = 0, %cfltop%                                            0252
    nambottom = 0, %namtop + namvinc%                               0253
    namcsize = 1, %character size%                                  0254
    namfont = 0, %font is normal%                                   0255
    namhinc = 0, %14*256%                                           0256
    namvinc = 0, %30*256%                                           0257
    namx = 0,                                                       0258
    namy = 0,                                                       0259
%subsystem display%                                                 0260
    subbase = subleft,                                              0261
    subleft = 0, %rmarg-16*subhinc%                                 0262
    subright = 0, %rmarg%                                           0263
    subtop = 0, %tmarg +  subvinc%                                  0264
    subbottom = 0, %subtop + subvinc%                               026
    subcsize = 1, %character size%                                  0266
    subfont = 0, %font is normal%                                   0267
    subhinc = 0, %14*256%                                           0268
    subvinc = 0, %30*256%                                           0269
    subx = 0,                                                       0270
    suby = 0,                                                       0271
%L-T view spec area (x=0, y=17008)%                                 0272
    ltbase = ltleft,                                                0273
    ltleft = 0, %lmarg%                                             0274
    ltright = 0, %ltleft+15*lthinc%                                 0275
    lttop = 0, %tmarg + ltvinc%                                     0276
    ltbottom = 0, %lttop + ltvinc%                                  0277
    ltcsize = 0, %character size%                                   0278
    ltfont = 0, %font is normal%                                    0279
    lthinc = 0, %12*256%                                            0280
    ltvinc = 0, %30*256%                                            0281
    ltx = 0,                                                        0282
    lty = 0,                                                        0283
%view spec area (x=0, y=1636)%                                      0284
    vsbase = vsleft,                                                0285
    vsleft = 0, %ltleft%                                            0286
    vsright = 0, %ltright%                                          0287
    vstop = 0, %ltbottom + ltvinc%                                  0288
    vsbottom = 0, %vstop + vsvinc%                                  0289
    vscsize = 0, %character size%                                   0290
    vsfont = 0, %font is normal%                                    0291
    vshinc = 0, %12*256%                                            0292
    vsvinc = 0, %30*256%                                            0293
    vsx = 0,                                                        0294
    vsy = 0,                                                        0295
%Literal feedback area%                                             0296
    litbase = litleft,                                              0297
    litleft = 0, %taleft%                                           0298
```

```
        litright = 0, %taright%                                          0299
        littop = 0, %tatop - 3*tavinc%                                   0300
        litbottom = 0, %tabottom%                                        0301
        litcsize = 1, %character size%                                   0302
        litfont = 0, %font is normal%                                    0303
        lithinc = 0, %14*256%                                            0304
        litvinc = 0, %30*256%                                            0305
        litx = 0,                                                        0306
        lity = 0,                                                        0307
    %Message area%                                                       0308
        msgbase = msgleft,                                               0309
        msgleft = 0, %cflleft%                                           0310
        msgright = 0, %dtleft -  msghinc%                                0311
        msgtop = 0, %tmarg + tavinc%                                     0312
        msgbottom = 0, %msgtop + msgvinc%                                0313
        msgcsize = 1, %character size%                                   0314
        msgfont = 0, %font is normal%                                    0315
        msghinc = 0, %14*256%                                            0316
        msgvinc = 0, %30*256%                                            0317
        msgx = 0,                                                        0318
        msgy = 0;                                                        0319
DECLARE EXTERNAL STRING %viewspecs%                                      0320
   vsstr1[10],      %L and T viewspec string%                            0321
   vsstr2[10];      %hjuCP viewspec string%                              0322
DECLARE EXTERNAL %line processor specific%                               0323
   dspjfn,              %JFN for line processor display%                 0324
   ldspjfn,             %dspjfn for linked display%                      0325
   lpbaudfactor=1,      %baud rate factor -- MCS-4 to terminal%          0326
   lpxmax=79,           %max x for line processor display%               0327
   lpymax=26,           %max y for line processor display%               0328
   lptype,              %type of line processor display%                 0329
   lpdlpad,             %delete line padding for line processor display%
                                                                         0330
   lptab = 40B,         %value of char to be shown as TAB%               0331
   lppadchr = 177B,     %value of char to be used for delays%            0332
   putbackchar=0,                                                        0333
      %TRUE if line-processor needs to have character rewritten when a
      bug mark is removed%                                               0334
   tracking,            %TRUE if line-processor is tracking mouse%       0335
   tracksend,  % count of  LP printer PSI's while not tracking %        0336
   lppch,         % pointer to lppch routine for given terminal type %
                                                                         0337
   lppcol,        % current column position for printer %               0338
   lpplin,        % current line number for printer %                   0339
   lppform,       % simulate formfeed flag %                           0340
   lpptimout,     % timeout cell for printer %                          0341
   lpppad,        % pad count for LF or what have you %                 0342
   lppjfn=0;      % jfn to copy to lp printer %                         0343
   DECLARE EXTERNAL STRING padstr =
   "°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°↑M
   °°°°°°°°°°°°°°°°°°°°";                                  0346
% declarations for use with the syntax genrating commands %              0403
   % global indicating current state of keyword recognition %            0414
      DECLARE EXTERNAL kwrstate;                                         0415
         % < 0  ==> have recognized the keyword associated with the
         last entry on the path stack %                                  0416
```

```
        % = 0  ==> have just entered keyword recognition mode, but
        have not recognized anything yet %                              0417
        % > 0  ==> have partially recognized the keyword associated
        with the last entry on the path stack %                         0418
    % this is storage for building a CML rule dynamically and for
    remembering how the rule was built %                                0404
       DECLARE EXTERNAL                                                 0405
          synsubs[ 400], synstore[2], acursub, asupsub, syncur;         0406
    % stack for  post-processing of unexpanded rules %                  0407
       DECLARE EXTERNAL pstkp = 0, pstk[50];                            0408
    % which commands do we get %                                        0409
       DECLARE EXTERNAL cmdctl = 0;                                     0410
          % 0 - all commands %                                          0411
          % 1 - all DNLS commands %                                     0412
          % 2 - all TNLS commands %                                     0413
% declarations used in INPFDBK %                                       0422
    % symtflg IF true the cntrlgetchar simulates recorded timing when
    playing back a record file set in xplayback%                        0423
       DECLARE EXTERNAL symtflg = 0;                                    0424
FINISH of fdata                                                         0347
```

FDBK.

(KIRK)FDBK (KIRK)FDBK (KIRK)FDBK (KIRK)FDBK (KIRK)FDBK (KIRK)F
(KIRK)FDBK (KIRK)FDBK (KIRK)FDBK (KIRK)FDBK (KIRK)FDBK (KIRK)F

< FEEDBACK, FDBK.NLS;220, >, 3-OCT-74 23:53 KIRK ;;;;

Introduction, Assumptions, Decision Algorithm 1

Assumptions and Goals: 1A

This file was created as part of an attempt to develop a methodology for collecting, storing, retrieving, analyzing, and making decisions concerning user feedback. This methodology has not been finalized but is described in its present form below. 1A1

It has generally been observed that a "suggestion box" capability is needed to take advantage of user feedback that will allow the user to be easily informed of the status of his suggestion and to provide some scales of measurement for analysis and development decisions. The algorithm below is a way of augmenting this process with the specific intent of avoiding any extra burden on the already over-burdened people involved. 1A2

There are several steps in the following algorithm. These have been customized for the internal ARC feedback process concerning software development and documentation. However, they could be generalized for any development group that wishes to augment their feedback process. 1A3

Each Development group could have it's own feedback ident, sndmessage directory, decision file, and person responsible for coordinating the file. For example: 1A4

GROUP	COORDINATOR	IDENT(S)	FILE	SNDMSG DIRECTORY
NLS DEVELOPMENT	JOAN	FDBK	<FEEDBACK,FDBK,>	FEEDBACK@SRI-ARC
USER DEVELOPMENT	JHB	FEED	<FEEDBACK,FEED,>	FEEDBACK@OFFICE-1
NIC DEVELOPMENT	JAKE	NIC	?	

1A5

If one group is sent an item which does not belong to that group, the feedback coordinator would be responsible for forwarding the item to the proper group. 1A6

A central Query Database could provide places from which any user could access the decision file to determine the status of his suggestion or to automatically express his opinion (via the Insert opinion command described below) concerning a development thrust as depicted by a decision file. 1A7

An ANALYSIS team uses the decision files to analyze the feedback and make recommendations. 1A8

It is assumed that the process of organizing, analyzing, and keeping track of user feedback, bugs, needs, possibilities, and thinkpieces is important enough to the decision process in a living system to warrant the allocation of the necessary initial setup and continuing maintenance time. 1A9

Algorithm for the NLS Software Development And Utility User Development Feedback Mechanism: 1B

USERS: 1B1

Send comments through SNDMSG to FEEDBACK@SRI-ARC or send them through the Journal to the ident: FDBK. 1B1A

Utility members send comments to their architect, who will forward them, or send them directly to ident FEED or through SNDMSG to FEEDBACK@OFFICE-1. An architect can send things to be shared by other architects to KWAC. 1B1B

Review the newly updated Future Needs & Possibilities branch when sent (see last step). They send their opinions to FDBK, use the "Insert Link" command or use the Insert Opinion Command (see SOFTWARE below). 1B1C

FEEDBACK COORDINATOR periodically (weekly): 1B2

Forwards items that belong to other groups to the appropriate ident. 1B2A

Reviews the Unclassified Items branch in <FEEDBACK>FDBK and makes an attempt to move bugs, trivia, Needs and Possibilities, and implemented items to the appropriate places and consolidates duplications. 1B2B

Prints out (or otherwise gives to DEVELOPMENT) the BUGS and ACCEPTED TASKS branches. 1B2C

NLS SOFTWARE DEVELOPMENT: 1B3

Once a week, or as provided by the Feedback Coordinator, reviews each item in the BUGS and ACCEPTED TASKS branches, and marks it as 1B3A

Future NP
Done
Rejects it or
Assigns it to an implementer. 1B3A1

* Moves marked items to their assigned branches in FDBK. 1B3B

Bug reports which are not clear enough should be rejected and the sender informed. 1B3B1

[An optional step depending on the available time allocated for Feedback purposes would be to forward this current status information to the user(s) who made the suggestion.] 1B3B2

[items to be completed by the Feedback coordinator.]* 1B3B3

Updates the Accepted tasks branch in the Feedback file to contain the current NLS implementation categories with the ident of the implementer responsible for each category. 1B3C

To items moved into the DONE branch, adds any changes made that

are different from or not included in user feedback with descriptions sufficiently accurate for the documenter to document them. Note: if an implementer makes a change in the system and does not somehow get it represented in the DONE branch, IT WILL NOT BE DOCUMENTED. 1B3D

Notifies the Feedback Coordinator when bringing up a new system. 1B3E

After bringing up a new version of NLS, reviews each item in it's Future Needs & Possibilities branch and either 1B3F

confirms its current position,
moves it to a more appropriate place,
rejects it, or
assigns it to an implementer. 1B3F1

ANALYSIS: 1B4

Analyzes information in FDBK and integrates it with other data for the purpose of making suggestions to operations, NIC, user-development, help-development, software-development, and analysis and for determining problem areas to be followed up with further studies. Analyzes feedback mechanism for possible improvements. 1B4A

DOCUMENTATION: 1B5

Checks the Help System branch under Accepted Tasks and implements or rejects suggested changes. 1B5A

Periodically checks for new items in Done Branch, moves each item out of the branch into the Documented branch, and updates the documentation if necessary. <See -- documentation, manual,>. 1B5B

FEEDBACK COORDINATOR for each new version of NLS: 1B6

Places items in each development area in the Future Needs & Possibilities branch in a suggested order of execution as directed by ANALYSIS supported by analysis' data and suggestions if any. 1B6A

Whenever a new version of NLS comes up, the feedback coordinator journalizes the entire feedback file at this time for posterity and to save space. 1B6B

First, use SENDMAIL's Number Assign command to get the number for the file. 1B6B1

Then fill out the proper NLS version number and page total in the following form and process it. 1B6B2

TITLE: User Feedback Decisions leading to NLS-8.3
AUTHOR(S): FDBK
DISTRIBUTE FOR INFO-ONLY TO: SRI-ARC KWAC

COMMENT: This document contains the status of user feedback decisions for NLS-8.3. It is over 50 pages long, we advise you NOT to print it. Read it online. For the new features and bug fixes, see the Documented branch. For those suggestions that have been rejected, see the Rejected branch. The items scheduled to be done in the next version are in <NLS,MODS,>. Those items which remain as Needs & Possibilities are in <FEEDBACK,FDBK,FUTURE>.

1B6B2A

After journalization, items for that NLS version in the Executed and Rejected branches are deleted and the link in the branch statement for these branches is updated to point to the newly journalized file. 1B6C

Very old Future NP items should be verified with their suporters to determine if they are still desired. 1B6D

SOFTWARE to augment this Algorithm: 1C

Insert Opinion B/A: (in favor?) Y/N: OK:
This command would be in the readmail subsystem and would allow anyone to insert their initials in front of an item classified in FDBK independent of whether they have write access. A minus sign would preceed the idents of those that said "no". 1C1

The Insert Link command in Sendmail subsystem should be implemented to allow anyone to use it on an item classified in FDBK independent of whether they have write access. 1C2

Complete Feature B/A (exactly as stated?) Y/N: OK:
This command would move an item specified by an implementer from the implementer's "to do" branch to the appropriate category in the Implemented branch in FDBK to be documented. 1C3

Author: announcements of old versions of feedback 1D

FDBK 25-SEP-74 14:16 24054
User Feedback Decisions leading to NLS-8.1
Message: <HJOURNAL, 24051,> contains the status of user feedback decisions for NLS-8.1. It is over 100 pages long, we advise you NOT to print it. Read it online. For the new features and bug fixes, see the Documented branch. For those suggestions that have been rejected, see the Rejected branch. The items scheduled to be done in the next version are in <NLS,MODS>. Those items which remain as Needs & Possibilities are in <feedback,fdbk,future>.
*****Note: A UTHOR Copy*****

1D1

FDBK 24-SEP-74 23:37 24051
User Feedback Decisions leading to NLS-8.1
Location: (HJOURNAL, 24051, 1:w)
*****Note: Author Copy*****

1D2

First journali&ed fdbk file -- <24048,> 1D3

FILMNP

FILMNP FILMNP FILMNP FILMNP FILMNP FILMNP FILMNP FILMNP
FILMNP FILMNP FILMNP FILMNP FILMNP FILMNP FILMNP FILMNP

```
< NLS, FILMNP.NLS;135, >, 27-SEP-74 09:56 CHI ;;;;
FILE filmnp  % L10 <REL-NLS>filmnp %% (L10,) (rel-nls,filmnp.rel,) %  02
%.....FILE MANIPULATION SUPPORT ROUTINES.....%                        03
   %Declarations%
                                                                      05
      DECLARE                                                         06
         newer = 0, older = 1, chkpcf = TRUE,                         07
         ckp1 = 2, ckp2 = 3, pc = 1;                                  08
      DECLARE STRING %for 5- to 7-bit character translation%          09
         trn5t7 = " ABCDEFGHIJKLMNOPQRSTUVWXYZ23456";                 010
      RGGISTER                                                        011
         r1=1, r2=2, r3=3, p=7, wa=8, s=9, m=10, rp=11, a1=12,        012
         a2=13, a3=14, a4=15;                                         013
   %File Block Manipulation Stuff%
                                                                      026
      (lodrfb) %Load random file block.  Arguments are                027
         (1) file block number,                                       028
         (2) block type,                                              029
            or negative number if just want to get a core page,       030
         (3) file number.                                             031
         Returns page index (i.e. the index in CRPGAD and CORPST      032
         for the page).  Look in CRPGAD indexed to get the page       033
         address.  This is NOT called to load the header of a         034
         file -- rdhdr does that job.%                                035
         PROCEDURE (nblk, btype, fileno);                             036
         LOCAL                                                        037
            pgindex, %file block page index%                          038
            blk,     %block address of the page%                      039
            fl,      %file list header%                               040
            jfn,     %file number for jsyses%                         041
            access,  %read/write access to the page%                  042
            stent,   %pointer to status table entry%                  043
            ct;      %used as pointer into CORPST%                    044
         REF ct, blk, fl, stent;                                      045
         IF nxpg NOT IN [rfpmin,rfpmax] THEN nxpg ← rfpmin;           048
         %choose the next file block page to use%                     049
         &ct ← $corpst + pgindex ← nxpg;                              050
         % look for an empty page %                                   051
         WHILE ct.ctfull DO                                           052
            BEGIN                                                     053
            IF (pgindex ← pgindex+1) > rfpmax THEN                    054
               &ct ← $corpst + pgindex ← rfpmin                       055
            ELSE BUMP &ct;                                            056
            IF pgindex = nxpg THEN                                    057
               BEGIN %have checked all the pages. none empty%         058
               %search for an unfrozen page%                          059
               WHILE ct.ctfroz DO BEGIN                               060
                  IF (pgindex ← pgindex+1) > rfpmax THEN              061
                     &ct ← $corpst + pgindex ← rfpmin                 062
                  ELSE BUMP &ct;                                      063
                  IF pgindex = nxpg THEN rerror(); %all frozen%       064
                  END;                                                065
               IF ct.ctfile > 0 THEN %does belong to some file%       066
                  BEGIN                                               067
                  %revise the appropriate status table%               068
                  &stent ← filehead[ct.ctfile] + $rfbs - $filhed      069
```

```
            + ct.ctpnum;                                                    070
            stent.rfcore ← 0;                                               071
            END;                                                            072
         ct.ctfull ← FALSE;                                                 073
         END;                                                               074
      END;                                                                  075
   IF (nxpg ← pgindex+1) > rfpmax THEN nxpg ← rfpmin;                       076
   %set up corpst for the new page%                                         077
   ct ← 0;                                                                  078
   ct.ctfull ← TRUE;                                                        079
   IF btype < 0 THEN %just gexting a page%                                  080
      BEGIN                                                                 081
      %remove the page that is there now%                                   082
      r2.PH ← 4B5;                                                          084
      r2.RH ← crpgad[pgindex] / 512;                                        085
      !pmap( -1, r2, 0);                                                    087
      RETURN (pgindex);                                                     088
      END;                                                                  089
   ct.ctfile ← fileno;                                                      090
   ct.ctpnum ← nblk;                                                        091
   &stent ← filehead[fileno] + $rfbs - $filhed + nblk;                      092
   &fl ← (fileno - 1)*filstl + $filst;                                      093
   IF btype = niltyp AND fl.flpart OR stent.rfpart #then
      begin
      jfn ← fl.flpart;                                                      096
      access ← 14B10; %read-write%                                          097
      END                                                                   098
   ELSE                                                                     099
      BEGIN                                                                0100
      jfn ← fl.florig;                                                     0101
      access ← 1B11; %read only%                                           0102
      END;                                                                 0103
   IF jfn = 0 THEN                                                         0104
      BEGIN                                                                0105
      ct ← 0;                                                              0106
      IF NOT fl.flexis THEN err($"fst entry nonexistant");                 0107
      err($"Illegal JFN in LODRFB");                                       0108
      END;                                                                 0109
   %set up registers for PMAP%                                             0110
   &blk ← crpgad[pgindex];                                                 0111
   r1.LH ← jfn;                                                            0112
   r1.RH ← nblk;                                                           0117
   r2.LH ← 4B5;                                                            0114
   r2.RH ← &blk / 512;                                                     0115
   !pmap(r1, r2, access);                                                  0117
   stent.rfcore ← pgindex;                                                 0118
   IF btype = niltyp THEN %do the initialization%                          0119
      BEGIN                                                                0120
      IF fl.flpart THEN stent.rfpart ← TRUE;                               0121
      blk.fbpnum ← nblk;                                                   0122
         %rest is done by the calling procedure%                           0123
      END                                                                  0124
   ELSE %verify%                                                           0125
      IF blk.fbtype # btype OR blk.fbpnum # nblk THEN                      0126
         badfil(fileno);                                                   0127
   RETURN (pgindex) END.
```

```
                                                                    0128
   (badfil) %called when find something screwed up in the file%     0129
      PROCEDURE (fileno);                                           0130
      bfilno ← fileno; %save it for SIGNAL%                         0131
      SIGNAL(-5, $"Bad File");                                      0132
      END.                                                          0133
   (frzrfb) %This routine is called for all freezing and thawing of
   random file blocks.  Arguments are                               0134
      (1) file number,                                              0135
      (2) block number in the file,                                 0136
      (3) a 1 to freeze the block, and a -1 to thaw it.             0137
      RERROR is called if that block is not in core.  Anyone        0138
      who freezes a block must be sure it is thawed -- and          0139
      only once.%
                                                                    0140
      PROCEDURE (fileno, blkn, fr);                                 0141
      LOCAL                                                         0142
         pgindex, %file block page counter%                         0143
         ct;  %pointer to CORPST entry%                             0144
      REF ct;                                                       0145
      pgindex ← 1;                                                  0146
      &ct ← $corpst + 1;                                            0147
      DO                                                            0148
         BEGIN                                                      0149
         IF ct.ctfull AND                                           0150
            ct.ctfile = fileno AND                                  0151
            ct.ctpnum = blkn THEN                                   0152
               BEGIN                                                0153
               IF ct.ctfroz + fr NOT IN [0,7] THEN                  0154
                  err($"Block frozen too many times in FRZRFB");
                                                                    01500
               ct.ctfroz ← ct.ctfroz + fr;                          0155
               RETURN;                                              0156
               END;                                                 0157
         BUMP &ct;                                                  0158
         END                                                        0159
      UNTIL (pgindex ← pgindex+1) > rfpmax;                         0160
      err($"Block not found in FRZRFB");                            0161
      END.                                                          0162
                                                                    0163
   (frzblk) %Freeze block given as arguments                        0164
      (1) the index in CRPGAD of the block, and                     0165
      (2) the 1 or -1 for freeze or thaw.%                          0166
                                                                    0167
      PROCEDURE (pgindx, fr);                                       0168
      LOCAL ct; REF ct;                                             0169
      &ct ← $corpst + pgindx;                                       0170
      IF ct.ctfroz + fr NOT IN [0,7] THEN                           0171
         err($"Block frozen too many times in frzblk");            01501
      ct.ctfroz ← ct.ctfroz + fr;                                   0172
      RETURN END.                                                   0173
                                                                    0174
%Ring Utility Routines%
                                                                    0175
   (lodrng)  %This routine loads a ring block into core.  The STID is
   provided as argument.  Returns two results:                      0176
```

```
    (1) the CRPGAD index for the new page                                0177
    (2) the address of the ring element.%
                                                                         0178
    PROCEDURE (stid);                                                    0179
    LOCAL wc, pgindx, rn;                                                0180
    REF rn;                                                              0181
    % this needs to be fast -- so we'll cheat and get the fields
    out of the stid without using byte pointers.  the record
    definitions for stid in utilty have a comment to the effect
    that we are doing this. %                                            0184
    r2 ← stid;                                                           0185
    !HLRZ r1,r2;                                                         0186
    %the stfile field is right justified in the left halfword%           0187
    !HRRZ r2,r2;  %clear the left half%                                  0188
    !IDIVI r2,1B3; %split out the block number and the word count%
                                                                         0189
    wc ← r3;  % the rngblk number is in r2 %                             0190
    &rn ← filehead[r1] + $rngst - $filhed + r2;                          0191
    %load if necessary%                                                  0192
    IF (pgindx ← rn.rfcore) = 0 THEN %must load it%                      0193
       pgindx ← lodrfb(r2+rngbas, rngtyp, r1);                           0194
    RETURN (pgindx, crpgad[pgindx]+wc) END.
                                                                         0195
(goodrng) PROCEDURE(stid);                                               0196
    %Returns TRUE iff stid points to a ring element which is in
    use%                                                                 0197
    %assumes that the file number in the stid is ok%                     0198
    LOCAL rngblk, wc, rn;                                                0199
    REF rn;                                                              0200
    %split the psid into block number and word count%                    0201
    wc ← stid.stwc;                                                      0202
    rngblk ← stid.stblk;                                                 0203
    &rn ← filehead[stid.stfile] + $rngst - $filhed + rngblk;             0204
    %check that block number is legal%                                   0205
    RETURN(rngblk IN [0,rngm) AND rn.rfexis AND getsid(stid)#0);
                                                                         0206
    END.
                                                                         0207
(newrng) %find room for a new ring element and allocate it.
Called with file number of file where want the new element.
Returns                                                                  0208
    (1) STID and                                                         0209
    (2) the address of the new element.%                                 0210
    PROCEDURE (fileno);                                                  0211
    LOCAL                                                                0212
       rngblk, %counter for ring blocks%                                 0213
       rngtry, %possible block for use next%                             0214
       pgindx, %page number of where loaded%                             0215
       blkad,  %address of block where will allocate%                    0216
       fileh,  %location of file header%                                 0217
       nringl, %number of ring elements in block%                        0218
       freep,  %free list pointer in block%                              0219
       rngsta, %address of RNGST for the file%                           0220
       rngloc, %address of RNGL for the file%                            0221
       stid,   %new STID%                                                0222
       rn,     %pointer into the RNGST for the file%                     0223
```

```
   fl;     %Address of file status table entry%                    0224
REF rn, fl;                                                        0225
rngsta ← $rngst + fileh ← filehead[fileno] - $filhed;              0226
rngloc ← fileh + $rngl;                                            0227
%first check the ring block from which last allocated              0228
or freed an element (number saved in RNGLST)%                      0229
&rn ← rngsta + rnglst;                                             0230
IF rnglst IN [0,rngm) AND                                          0231
   rn.rfexis AND rn.rfcore AND rn.rffree THEN                      0232
   BEGIN                                                           0233
   stid ←                                                          0234
      nwrngb(fileh, rnglst, rn.rfcore, fileno : rngloc);           0235
   RETURN (stid, rngloc);                                          0236
   END;                                                            0237
%else search for a block%                                          0238
rngblk ← 0;                                                        0239
rngtry ← -1;                                                       0240
&rn ← rngsta;                                                      0241
DO BEGIN                                                           0242
   IF rn.rfexis AND rn.rffree THEN                                 0243
      IF rn.rfcore THEN                                            0244
         BEGIN                                                     0245
         stid ←                                                    0246
            nwrngb(fileh, rngblk, rn.rfcore, fileno :              0247
            rngloc);                                               0248
         RETURN (stid, rngloc);                                    0249
         END                                                       0250
      ELSE rngtry ← rngblk;                                        0251
   BUMP &rn;                                                       0252
   END                                                             0253
UNTIL (rngblk ← rngblk+1) > [rngloc];                              0254
IF rngtry >= 0 THEN %load one that has room%                       0255
   BEGIN                                                           0256
   stid ← 0;                                                       0257
   stid.stfile ← fileno;                                           0258
   stid.stblk ← rngtry;                                            0259
   pgindx ← lodrng(stid : blkad);                                  0260
   stid ← nwrngb(fileh, rngtry, pgindx, fileno : rngloc);          0261
   RETURN (stid, rngloc);                                          0262
   END;                                                            0263
%must allocate a new block%                                        0264
rngblk ← 0;                                                        0265
&rn ← rngsta;                                                      0266
DO BEGIN                                                           0267
   IF NOT rn.rfexis THEN %will initialize this block%              0268
      BEGIN                                                        0269
      IF NOT filepart[fileno] THEN %Make pc if neccessary%         0270
         BEGIN                                                     0271
         &fl ← (fileno-1)*filstl + $filst;                         0272
         IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT
         makepc(&fl, fileno, $lit) THEN                            0273
            BEGIN                                                  0274
            IF NOT fl.flbrws THEN lkun(&fl, $lit); %change user
            setable word to unlock%                                0275
            dismes(2, $lit);                                       0276
            GOTO STATE;                                            0277
```

```
            END;                                                    0278
          END;                                                      0279
        pgindx ← lodrfb(rngbas+rngblk, niltyp, fileno);             0280
        blkad ← crpgad[pgindx];                                     0281
        %finish initialization of the header%                       0282
        [blkad].fbtype ← rngtyp;  [blkad].fbind ← rngblk;           0283
        %now set up the status table%                               0284
        rn.rfexis ← TRUE;                                           0285
        rn.rfused ← rn.rffree ← fbhdl;                              0286
        rn.rfcore ← pgindx;                                         0287
        %update RNGL for the file%                                  0288
        IF rngblk > [rngloc] THEN [rngloc] ← rngblk;                0289
        %finish the initialization of the block%                    0290
        %make a free list%                                          0291
        %calculate the number of elements that will fit             0292
        in a page%                                                  0293
        nringl ← (blksiz-fbhdl) / ringl;                            0294
        freep ← blkad + fbhdl;                                      0295
        DO BEGIN                                                    0296
           [freep] ← freep - blkad + ringl;                         0297
           freep ← freep + ringl                                    0298
           END                                                      0299
        UNTIL (nringl ← nringl-1) = 1;                              0300
        %zero the last one%                                         0301
        [freep] ← 0;                                                0302
        stid ← nwrngb(fileh, rngblk, pgindx, fileno :               0303
        rngloc);                                                    0304
        RETURN ( stid, rngloc);                                     0305
        END;                                                        0306
     BUMP &rn;                                                      0307
     END                                                            0308
  UNTIL (rngblk ← rngblk+1) = rngm;                                 0309
  %have exhausted the space available for structure%                0310
  err($"structure full") END.                                       0311

(nwrngb) %Used by newrng to actually allocate the ring element.
Generates new SID and stores it in the ring element.  Called with   0312

  (1) address of file header - $filhed,                             0313
  (2) the ring block number,                                        0314
  (3) the page index of the ring block,                             0315
  (4) the file number.                                              0316
  It allocates a new ring element from that block and returns       0317

  (1) the new statement identifier (STID) and                       0318
  (2) the address of the new element.%                              0319
  PROCEDURE (fileh, rngblk, pgindx, fileno);                        0320
  LOCAL                                                             0321
     rn,      %address of ring entry in header%                     0322
     freep,   %free list pointer%                                   0323
     nwrn,    %pointer to new ring block%                           0324
     nwrne,   %pointer to end of new ring block%                    0325
     stid,    %stid for the new element%                            0326
     fl;      %Address of file status table entry%                  0327
  REF rn, fl;                                                       0328
  %record the block number from which allocating the                0329
```

```
    element%                                                        0330
    rnglst ← rngblk;                                                0331
    &rn ← fileh + $rngst + rngblk;                                  0332
    %check the free pointer for legality%                           0333
    IF (freep ← rn.rffree)                                          0334
       NOT IN [fbhdl,blksiz) THEN badfil(fileno);                   0335
    IF NOT filepart[fileno] THEN %Make pc if neccessary%            0336
       BEGIN                                                        0337
       &fl ← (fileno-1)*filstl + $filst;                            0338
       IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT makepc(&fl,
       fileno, $lit) THEN                                           0339
          BEGIN                                                     0340
          IF NOT fl.flbrws THEN lkun(&fl, $lit); %change user
          setable word to unlock%                                   0341
          dismes(2, $lit);                                          0342
          GOTO STATE;                                               0343
          END;                                                      0344
       END;                                                         0345
    freep ← freep + crpgad[pgindx]; %actual address%                0346
    rn.rffree ← [freep]; %new free pointer%                         0347
    rn.rfused ← rn?rfused + ringl; %increase used word count%       0348
    %zero the new ring element%                                     0349
    nwrne ← (nwrn ← freep) + ringl;                                 0350
    DO [nwrn] ← 0 UNTIL (nwrn ← nwrn+1) = nwrne;                    0351
    % get new SID %                                                 0352
    [freep].rsid ← [fileh + $sidcnt] ← [fileh + $sidcnt]            0353
    + 1;                                                            0354
    %return STID for the new element%                               0355
    stid ← 0;                                                       0356
    stid.stfile ← fileno;                                           0357
    stid.stblk ← rngblk;                                            0358
    stid.stwc ← freep-crpgad[pgindx];                               0359
    [freep].rsub ← stid;                                            0360
    RETURN (stid, freep);                                           0361
    END.                                                            0362

(frerng)  %free the ring element for STID given as argument%
                                                                    0363
    PROCEDURE (stid);                                               0364
    LOCAL                                                           0365
       rngloc, %location of the ring element%                       0366
       blkad,  %address of file block containing the element%       0367
       pgindx, %index of ring block%                                0368
       cnt,    %counter for clearing element%                       0369
       pnt,    %pointer for clearing element%                       0370
       rn,    %pointer to RNGST entry%                              0371
       fl;     %Address of file status table entry%                 0372
    REF rn, fl;                                                     0373
    %zap swork if necessary%                                        0374
       IF swork = stid THEN swork ← endfil;                         0375
       %this is done for the benefit of fechcl.  ensures that when
       a FIND fails and CCPOS is reset to position before the FIND,
       it will be reset to an existing statement or to the NULL
       string.%                                                     0376
    IF NOT filepart[stid.stfile] THEN %Make pc if neccessary%       0377
       BEGIN                                                        0378
```

```
        &fl ← (stid.stfile-1)*filstl + $filst;                          0379
        IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT makepc(&fl,
        stid.stfile, $lit) THEN                                         0380
           BEGIN                                                        0381
           IF NOT fl.flbrws THEN lkun(&fl, $lit); %change user
           setable word to unlock%                                      0382
           dismes(2, $lit);                                             0383
           GOTO STATE;                                                  0384
           END;                                                         0385
        END;                                                            0386
     rnglst ← stid.stblk;                                               0387
     pgindx ← lodrng(stid : rngloc);                                    0388
     blkad ← crpgad[pgindx];                                            0389
     %clear the element%                                                0390
     cnt ← ring1;                                                       0391
     pnt ← rngloc;                                                      0392
     DO                                                                 0393
        BEGIN                                                           0394
        [pnt] ← 0;                                                      0395
        BUMP pnt;                                                       0396
        END                                                             0397
     UNTIL (cnt ← cnt-1) = 0;                                           0398
     &rn ← filehead[stid.stfile] + $rngst - $filhed +                   0399
     stid.stblk;                                                        0400
     %reduce used word count%                                           0401
     rn.rfused ← rn.rfused - ring1;                                     0402
     %add to free list%                                                 0403
     [rngloc] ← rn.rffree;                                              0404
     rn.rffree ← rngloc - blkad;                                        0405
     RETURN END.
                                                                        0406
%Store's with Ring Elements%
                                                                        0407
   (stosuc) %Store successor of specified statement.%
                                                                        0408
      PROCEDURE (stid, sucstid);                                        0409
      LOCAL rngloc;                                                     0410
      lodrng(stid : rngloc);                                            0411
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                        0412
      [rngloc].rsuc ← sucstid.stpsid;                                   0413
      RETURN END.
                                                                        0414
   (stosub) %Store sub of specified statement.%
                                                                        0415
      PROCEDURE (stid, substid);                                        0416
      LOCAL rngloc;                                                     0417
      lodrng(stid : rngloc);                                            0418
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                        0419
      [rngloc].rsub ← substid.stpsid;                                   0420
      RETURN END.
                                                                        0421
   (stoftl) %Store tail flag of specified statement.%
                                                                        0422
      PROCEDURE (stid, tail);                                           0423
```

```
    LOCAL rngloc;                                                   0424
    lodrng(stid : rngloc);                                          0425
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0426
    [rngloc].rtf ← tail;                                            0427
    RETURN END.
                                                                    0428
(stofhd) %store head flag of specified statement.%
                                                                    0429
    PROCEDURE (stid, head);                                         0430
    LOCAL rngloc;                                                   0431
    lodrng(stid : rngloc);                                          0432
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0433
    [rngloc].rhf ← head;                                            0434
    RETURN END.
                                                                    0435
(stonmf) %Store name flag of specified statement.%
                                                                    0436
    PROCEDURE (stid, name);                                         0437
    LOCAL rngloc;                                                   0438
    lodrng(stid : rngloc);                                          0439
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0440
    [rngloc].rnamef ← name;                                         0441
    RETURN END.
                                                                    0442
(stosdb) %Store the psdb of specified statement.%
                                                                    0443
    PROCEDURE (stid, stdb);                                         0444
    LOCAL rngloc;                                                   0445
    lodrng(stid : rngloc);                                          0446
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0447
    [rngloc].rsdb ← stdb.stpsdb;                                    0448
    RETURN END.
                                                                    0449
(stonam) %Store the name hash of specified statement.%
                                                                    0450
    PROCEDURE (stid, nameh);                                        0451
    LOCAL rngloc;                                                   0452
    lodrng(stid : rngloc);                                          0453
    IF [rngloc].rsd = 0 THEN err($"Bad statement identifier");
                                                                    0454
    [rngloc].rnameh ← nameh;                                        0455
    RETURN END.
                                                                    0456
(stosid) %Store the name hash of specified statement.%
                                                                    0457
    PROCEDURE (stid, sid);                                          0458
    LOCAL rngloc;                                                   0459
    lodrng(stid : rngloc);                                          0460
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0461
    [rngloc].rsid ← sid;                                            0462
    RETURN END.
```

```
                                                                    0463
%basic manipulation routines%
                                                                    01191
   (copgrp)                                                         01192
      %Given an stid as the first argument, this routine copies the
      group bounded by the second and third arguments after the first
      stid, in the direction specified by the fourth argument.  The
      fifth argument indicates whether the correspondence list should
      be updated.                                                   01193
         It proceeds by constructing new ring elements for each
         branch defined by the top-level statements in the group.
         Then it copies these data blocks into freshly allocated
         SDB's.  It then inserts the newly created group after the
         stid passed it.%                                           01194
      %-------------%                                              01195
      PROCEDURE(stid, dir, grp1, grp2, upctf);                      01196
      LOCAL newsid, %new stid%                                      01197
         oldsid, %old stid being processed%                         01198
         newgp1; %head of new group%                                01199
      IF grp1.stfile # grp2.stfile THEN err($"illegal group");      01200
      oldsid ← grp1;                                                01201
      newsid ← newgp1 ← newrng(stid.stfile);                        01202
      LOOP                                                          01203
         BEGIN                                                      01204
         %get stid's, this branch%                                  01205
         WHILE (oldsid := getsub(oldsid)) # oldsid DO               01206
            newsid ← newsub(newsid);                                01207
         %now copy sdb's and point to next branch%                  01208
         LOOP                                                       01209
            BEGIN                                                   01210
            copsdb(oldsid, newsid);                                 01211
            IF upctf THEN upctbl(oldsid, newsid, oldsid);           01212
            IF oldsid = grp1 THEN %branch copy complete%            01213
               BEGIN                                                01214
               IF oldsid = grp2 THEN %group copy complete%          01215
                  BEGIN                                             01216
                  insgrp(stid, dir, newgp1, newsid);                01217
                  RETURN(newsid);                                   01218
                  END;                                              01219
               oldsid ← grp1 ← getsuc(oldsid);                      01220
               EXIT;                                                01221
               END;                                                 01222
            IF NOT getftl(oldsid := getsuc(oldsid)) THEN EXIT;      01223
               %still more in plex%                                 01224
            newsid ← getsuc(newsid);                                01225
            END;                                                    01226
         newsid ← newsuc(newsid);                                   01227
         END;                                                       01228
      END.
                                                                    01229
   (copfgrp) PROCEDURE      % copy filtered group %                 01807
      (target, rlevcnt, src1, src2, vsptr);                         01808
      LOCAL retstid, newtgt, newsrc, lstlev, toplev, sqgwrk, vspec1,
      vspec2, usqcod, cacode;                                       01809
      LOCAL TEXT POINTER tp1, tp2;                                  02008
      REF sqgwrk, vsptr;                                            01810
```

```
                                                                    01811
% initialize %                                                      01812
   vspec1 ← vsptr.vs1;                                              01813
   vspec2 ← vsptr.vs2;                                              01814
   cacode ← vsptr.vscacode;                                         01815
   usqcod ← vsptr. vsusqcod;                                        01816
   retstid ← target;                                                01817
&sqgwrk ← openseq (src1, src2, vspec1, vspec2, usqcod, cacode);
                                                                    01818
ON SIGNAL ELSE IF &sqgwrk THEN                                      02003
   BEGIN                                                            02004
   ON SIGNAL ELSE;                                                  02005
   closeseq( &sqgWrk := 0 );                                        02006
   END;                                                             02007
IF (newsrc ← seqgen(&sqgwrk)) # endfil THEN                         01819
   BEGIN                                                            01820
   IF newsrc.stastr THEN                                            01822
      BEGIN                                                         02009
      FIND SF(newsrc) ↑tp1 SE(newsrc) ↑tp2;                         02010
      retstid ← newtgt ← cinssta( target, rlevcnt, $tp1, $tp2 )
                                                                    01821
      END                                                           02011
   ELSE retstid ← newtgt ←                                          01824
      ccopsta( target, rlevcnt, newsrc, FALSE, FALSE);              02016
   toplev ← lstlev ←                                                01825
      IF sqgwrk.swclvl = 0 THEN 1 ELSE sqgwrk.swclvl;               01826
   WHILE (newsrc ← seqgen(&sqgwrk)) # endfil DO                     01827
      BEGIN                                                         01828
      rlevcnt ← 0;                                                  01829
      CASE sqgwrk.swclvl OF                                         01830
         =lstlev: NULL;                                             01831
         >lstlev:                                                   01832
            BEGIN                                                   01833
            rlevcnt ← -1;                                           01834
            lstlev ← lstlev + 1;                                    01835
            END;                                                    01836
         <lstlev:                                                   01837
            WHILE lstlev > toplev DO                                01838
               BEGIN                                                01839
               BUMP rlevcnt;                                        01840
               IF ((lstlev ← lstlev - 1) = sqgwrk.swclvl) THEN
                                                                    01841
                  EXIT LOOP 1;                                      01842
               END;                                                 01843
         ENDCASE NULL;                                              01844
      IF newsrc.stastr THEN                                         01846
         BEGIN                                                      02012
         FIND SF(newsrc) ↑tp1 SE(newsrc) ↑tp2;                      02013
         newtgt ← cinssta( newtgt, rlevcnt, $tp1, $tp2 )            01845
         END                                                        02015
      ELSE newtgt ←                                                 01848
         ccopsta( newtgt, rlevcnt, newsrc, FALSE, FALSE);           02017
      END;                                                          01849
   END;                                                             01850
closeseq (&sqgwrk := 0);                                            01851
RETURN(retstid);                                                    01852
```

```
    END.
                                                                    01853
(mvdlfgrp)          % move / delete filtered group %                 01855
    PROCEDURE                                                       01856
       (stid1,      %stid of first statement to be moved/deleted%
                                                                    01857
       stid2,       %stid of second statement to be moved deleted%
                                                                    01858
       vsptr,       %pointer to filter viewspec record%             01859
       newstid,     %stid of statement following group: stid1 stid2%
                                                                    01860
       type,        %delete/move/transpose%                         01990
       mvdlflag,    %stid of target for move/transpose%             01861
       mvrlev       %relative level for move/transpose%             01862
       );                                                           01863
    LOCAL                                                           01864
       vspec1, unfseq, ufstid, ufring, filseq, filstid, filring,
       cring, pflag, rstid, toplev, lstlev, mtoplev, mlstlev,
       rlevcnt, cstid, nstid;                                       01865
    REF vsptr, unfseq, ufring, filseq, filring, cring;              01866
                                                                    01867
    vspec1 ← vsptr.vs1;                                             01868
       vspec1.vslev ← -1;  % all levels %                           01869
       vspec1.vscapf ← FALSE;  % no content analyzer program %
                                                                    01870
       vspec1.vsusqf ← FALSE;  % no sequence generator program %
                                                                    01871
    % make unfiltered pass thru group; mark all with absolute level
    %                                                               01872
       % open an unfiltered sequence %                             01873
          &unfseq ←                                                 01874
             openseq( stid1, stid2, vspec1, vsptr.vs2, FALSE,
             FALSE);                                                01875
       ON SIGNAL ELSE IF &unfseq THEN closeseq (&unfseq :=0);       01876
       WHILE (ufstid ← seqgen(&unfseq)) # endfil DO                 01877
          BEGIN                                                     01878
          lodrng( ufstid : &ufring);                                01879
          ufring.rinst1 ← unfseq.swc1v1;                            01880
          ufring.rdummy ← TRUE;                                     01881
          END;                                                      01882
       closeseq( &unfseq := 0 );                                    01883
    % make filtered pass marking all statements that pass false %
                                                                    01884
       % open a filtered sequence %                                01885
          &filseq ←                                                 01886
             openseq( stid1, stid2, vsptr.vs1, vsptr.vs2,
             vsptr.vsusqcod, vsptr.vscacode);                       01887
       ON SIGNAL ELSE IF &unfseq THEN closeseq (&unfseq :=0);       01888
       WHILE (filstid ← seqgen(&filseq)) # endfil DO                01889
          BEGIN                                                     01890
          lodrng( filstid : &filring);                              01891
          filring.rdummy ← FALSE;                                   01892
          END;                                                      01893
       closeseq( &filseq := 0 );                                    01894
       ON SIGNAL ELSE;                                              01895
    % get predessor or up (remember which) of first unfiltered %
```

```
                                                                    01896
   lodrng( stid1 : &cring );                                        01897
   pflag ← IF cring.rhf THEN FALSE ELSE TRUE;                       01898
   rstid ← IF pflag THEN getprd(stid1) ELSE getup(stid1);           01899
% remove the entire group %                                        01900
   IF NOT remgrp( stid1, stid2) THEN                                01901
      CASE type OF                                                  01991
         = moveflag: err(@"illegal move");                          01992
         = trnsflag: err(@"illegal transpose");                     01993
         = deltflag: err(@"illegal delete");                        01994
         ENDCASE;                                                   01995
% initialize level stuff %                                         01904
   toplev ← lstlev ← mtoplev ← mlstlev ← 0;                         01905
% initialize for main loop %                                       01906
   cstid ← stid1;                                                   01907
% final pass through removed group %                               01908
   WHILE cstid DO                                                   01909
      BEGIN                                                         01910
      % get ring element for current stid %                        01911
         lodrng( cstid : &cring );                                  01912
      % set up stack for next statement %                          01913
         IF cstid = stid2 THEN                                      01914
            BEGIN                                                   01915
            nstid ← 0;                                              01916
            !PUSH s,nstid                                           01917
            END                                                     01918
         ELSE IF NOT cring.rtf THEN                                 01919
            BEGIN                                                   01920
            nstid ← cring.rsuc;                                     01921
            nstid.stfile ← stid1.stfile;                            01922
            !PUSH s,nstid;                                          01923
            END;                                                    01924
         cring.rsuc ← 0;                                            01925
         IF (nstid← cring.rsub:= cstid.stpsid) # cstid.stpsid
         THEN                                                       01926
            BEGIN                                                   01927
            nstid.stfile ← stid1.stfile;                            01928
            !PUSH s,nstid;                                          01929
            END;                                                    01930
      IF cring.rdummy THEN                                          01931
         CASE type OF                                               01996
            = trnsflag: delgrp(cstid, cstid, newstid);              01997
            ENDCASE                                                 01998
               BEGIN  % insert the statement %                      01932
               IF NOT toplev THEN                                   01933
                  BEGIN                                             01934
                  toplev ← lstlev ← cring.rinst1;                   01935
                  rlevcnt ← IF pflag THEN 0 ELSE -1;                01936
                  END                                               01937
               ELSE rlevcnt ← 0;                                    01938
               CASE cring.rinst1 OF                                 01939
                  = lstlev:  NULL;                                  01940
                  > lstlev:                                         01941
                     BEGIN                                          01942
                     rlevcnt ← -1;                                  01943
                     BUMP lstlev;                                   01944
```

```
                        END;                                        01945
                     < lstlev:                                      01946
                        WHILE lstlev > toplev DO                    01947
                           BEGIN                                    01948
                           BUMP rlevcnt;                            01949
                           IF ((lstlev ← lstlev - 1) =
                           cring.rinst1) THEN                       01950
                              EXIT LOOP 1;                          01951
                           END;                                     01952
                     ENDCASE;                                       01953
                  insgrp( rstid := cstid, rlevcnt, cstid, cstid );
                                                                    01954
                  END                                               01955
            ELSE                                                    01956
               CASE type OF                                         02000
                  = deltflag: delgrp(cstid, cstid, newstid);        02001
                  ENDCASE                                           02002
                     BEGIN  % move the statement %                  01959
                     IF NOT mtoplev THEN                            01960
                        BEGIN                                       01961
                        mtoplev ← mlstlev ← cring.rinst1;           01962
                        rlevcnt ← mvrlev;                           01963
                        END                                         01964
                     ELSE rlevcnt ← 0;                              01965
                     CASE cring.rinst1 OF                           01966
                        = mlstlev:  NULL;                           01967
                        > mlstlev:                                  01968
                           BEGIN                                    01969
                           rlevcnt ← -1;                            01970
                           BUMP mlstlev;                            01971
                           END;                                     01972
                        < mlstlev:                                  01973
                           WHILE mlstlev > mtoplev DO               01974
                              BEGIN                                 01975
                              BUMP rlevcnt;                         01976
                              IF ((mlstlev ← mlstlev - 1) =
                              cring.rinst1) THEN                    01977
                                 EXIT LOOP 1;                       01978
                              END;                                  01979
                        ENDCASE;                                    01980
                     insgrp( mvdlflag := cstid, rlevcnt, cstid, cstid
                     );                                             01981
                     END;                                           01982
            % get next stid %                                       01983
               !POP s,cstid;                                        01984
            END;                                                    01985
      RETURN;                                                       01986
      END.                                                          01987
   (remgrp)                                                         01230
      %Removes group whose bounds are passed it from position in ring
      but does not delete either ring elements or SDB's.%           01231
      %-------------%                                               01232
      PROCEDURE(grp1, grp2);                                        01233
      LOCAL stid; %work area for updating stid%                     01234
      IF grp1.stpsid = origin OR                                    01235
         grp2.stpsid = origin THEN RETURN(FALSE);                   01236
```

```
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");  01237
   %putcdpanic(IF getfhd(grp1) THEN getup(grp1)              01238
      ELSE getprd(grp1));%                                   01239
   IF getfhd(grp1) THEN                                      01240
      BEGIN                                                  01241
      stid ← getsuc(grp2); %new plex head%                   01242
      IF NOT getft1(grp2) THEN stofhd(stid, TRUE);           01243
      stosub(getup(grp2), stid);                             01244
      END                                                    01245
   ELSE                                                      01246
      BEGIN                                                  01247
      stid ← getprd(grp1);                                   01248
      stoft1(stid, getft1(grp2));                            01249
      stosuc(stid, getsuc(grp2));                            01250
      END;                                                   01251
   RETURN;                                                   01252
   END.
                                                             01253
(delgrp)                                                     01254
   %This routine destroys all trace of the group bounded by the
   arguments passed it.  (Note that it does not remove the group
   from the ring; it assumes that this has been done by, for
   example, REMGRP.)                                         01255
      It follows each branch to its deepest level, deleting
      references to substructure in the sub fields; it then
      follows the structure back up, through the suc pointers,
      freeing SDB's as it goes.%                             01256
   %--------------%                                          01257
   PROCEDURE(grp1, grp2, newsid);                            01258
   LOCAL                                                     01259
      stid, %stid being processed%                           01260
      subsid; %sub of stid being processed%                  01261
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");  01262
   IF grp1.stpsid = origin OR                                01263
      grp2.stpsid = origin THEN err($"illegal delete");      01264
   stid ← grp1;                                              01265
   LOOP                                                      01266
      BEGIN                                                  01267
      WHILE (subsid ← getsub(stid)) # stid DO                01268
         BEGIN                                               01269
         stosub(stid, stid);                                 01270
         stid ← subsid;                                      01271
         END;                                                01272
      stid ← getsuc(subsid); %now go back up%                01273
      upctbl(subsid, endfil, newsid);                        01274
      <TXTEDT,dsttx> (subsid); %free sdb%                    01275
      freerng(subsid);                                       01276
      IF subsid = grp2 THEN RETURN;                          01277
      END;                                                   01278
   END.
                                                             01279
%correspondence table manipulation%
                                                             01459
   (upctbl)                                                  01460
      %Given three stid's, this routine will update occurrences of
      the first stid, using rplsid as the stid that contains the
```

```
        text corresponding to oldsid, and newsid as the stid that is
        the replacement for oldsid.  (rplsid should be endfil if the
        original text has been eliminated.)%                          01461
        %-------------%                                               01462
        PROCEDURE(oldsid, rplsid, newsid);                            01463
        LOCAL list, listnd;                                           01464
        REF list;                                                     01465
        IF clstad = 0 OR [clstad].clbuff = 0 THEN RETURN;             01466
        &list ← [clstad].clbuff;                                      01467
        listnd ← &list + [clstad].clcnt * cll;                        01468
        FOR &list UP cll UNTIL >= listnd                              01469
        DO                                                            01470
           IF list.clstl = oldsid THEN                                01471
              BEGIN                                                   01472
              list.clst2 ← newsid;                                    01473
              list.clstl ← rplsid;                                    01474
              END                                                     01475
           ELSE                                                       01476
              IF list.clst2 = oldsid THEN                             01477
                 list.clst2 ← newsid;                                 01478
        RETURN;                                                       01479
        END.
                                                                      01480
%structural inserts%
                                                                      01280
    (newsuc)                                                          01281
       %This routine gets a new stid and then inserts it as the suc of
       the stid passed it.  It returns the new stid.%                 01282
       %-------------%                                               01283
       PROCEDURE(stid);                                               01284
       LOCAL newstd; %new stid%                                       01285
       newstd ← newrng(stid.stfile);                                  01286
       inss(stid, newstd, newstd);                                    01287
       RETURN(newstd);                                                01288
       END.
                                                                      01289
    (newsub)                                                          01290
       %This routine gets a new stid and then inserts it as the sub of
       the stid passed it.  It returns the new stid.%                 01291
       %-------------%                                               01292
       PROCEDURE(stid);                                               01293
       LOCAL newstd; %new stid%                                       01294
       newstd ← newrng(stid.stfile);                                  01295
       insd(stid, newstd, newstd);                                    01296
       RETURN(newstd);                                                01297
       END.
                                                                      01298
    (insgrp)                                                          01299
       %Insert SRC1, SRC2 at TARGET according to DIR.  If DIR is true,
       then the group is inserted as the successor; otherwise it is
       inserted as the sub.  (SRC1 and SRC2 are assumed to define a
       legal, ordered group.)%                                        01300
       %-------------%                                               01301
       PROCEDURE(target, dir, src1, src2);                            01302
       target ← rlevset(target, dir : dir);                           01303
       IF target.stpsid # origin AND dir = levsuc THEN                01304
```

```
      inss(target, src1, src2)                                          01305
   ELSE insd(target, src1, src2);                                       01306
   RETURN;                                                              01307
   END.
                                                                        01308
(inss)                                                                  01309
   %Given an stid, this routine inserts a group, defined by the
   second and third arguments, as the suc of the first argument.
   First it makes the suc of STID the suc of GRP2, and updates the
   tail flag; then it makes GRP1 the suc of STID and updates the
   head and tail flags.%                                                01310
   %--------------%                                                     01311
   PROCEDURE(stid, grp1, grp2);                                         01312
   IF stid.stfile # grp1.stfile THEN err($"illegal insert");            01313
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");             01314
   stosuc(grp2, getsuc(stid));                                          01315
   stoftl(grp2, getftl(stid));                                          01316
   stosuc(stid, grp1);                                                  01317
   stofhd(grp1, FALSE);                                                 01318
   stoftl(stid, FALSE);                                                 01319
   RETURN;                                                              01320
   END.
                                                                        01321
(insd)                                                                  01322
   %Given an stid, this routine inserts the group defined by the
   second and third arguments down from the first argument.%            01323
   %--------------%                                                     01324
   PROCEDURE(stid, grp1, grp2);                                         01325
   LOCAL substd; %stid of sub of stid passed as argument%               01326
   IF stid.stfile # grp1.stfile THEN err($"illegal insert");            01327
   IF grp1.stfile # grp2.stfile THEN err($"illegal group");             01328
   IF (substd ← getsub(stid)) # stid THEN                               01329
      BEGIN                                                             01330
      stofhd(substd, FALSE);                                            01331
      stoftl(grp2, FALSE);                                              01332
      END                                                               01333
   ELSE stoftl(grp2, TRUE);                                             01334
   stosuc(grp2, substd);                                                01335
   stosub(stid, grp1);                                                  01336
   stofhd(grp1, TRUE);                                                  01337
   RETURN;                                                              01338
   END.
                                                                        01339
(rlevset)                                                               01340
   % Determines target stid and direction for inserting
   statements.  Given an stid and a relative levadj count, returns
   an stid and levdown if to be inserted down, levsuc if to be
   inserted as successor to returned stid. %                            01341
   %---------------%                                                    01342
   PROCEDURE(stid, dir);                                                01343
   LOCAL count, numb;                                                   01344
   CASE dir OF                                                          01345
      =0: dir ← levsuc; %successor%                                     01346
      <0: dir ← levdown; %down%                                         01347
      ENDCASE %up number of levels indicated by dir%                    01348
         WHILE dir > 0 DO                                               01349
```

```
            BEGIN                                                   01350
            stid ← getup(stid);                                     01351
            dir ← dir - 1;                                          01352
            END;                                                    01353
      RETURN(stid, dir);                                            01354
      END.
                                                                    01355
   (levset)                                                         01356
      % Determines target stid and direction for inserting
      statements.  Given an stid and the address of a string
      containing u's and d's (a levadj string), returns an stid and
      -1 if to be inserted down, 0 if to be inserted as successor to
      returned stid. %                                              01357
      %--------------%                                             01358
      PROCEDURE(stid, levstg);                                      01359
      LOCAL dir, count, numb;                                       01360
      REF levstg;                                                   01361
      dir ← 0;                                                      01362
      IF levstg.L # empty THEN                                      01363
         BEGIN                                                      01364
         count ← 1;                                                 01365
         DO                                                         01366
            CASE *levstg*[count] OF                                 01367
               ='U, ='u: BUMP DOWN dir;                             01368
               ='D, ='d: BUMP dir;                                  01369
               ENDCASE NULL                                         01370
         UNTIL (count ← count + 1) > levstg.L;                      01371
         END;                                                       01372
      CASE dir OF                                                   01373
         =0: BUMP dir; %successor%                                  01374
         >0: dir ← 0 %down%                                         01375
         ENDCASE %up number of levels indicated by dir%             01376
            WHILE (dir ← dir + 1) <= 0                              01377
               DO stid ← getup(stid);                               01378
                                                                    01379
      RETURN(stid, dir - 1);                                        01380
      END.
                                                                    01381
%move around in ring%
                                                                    01382
   (getail)                                                         01383
      %Given an stid, this procedure returns the stid of the tail of
      the current plex%                                             01384
      %--------------%                                             01385
      PROCEDURE (stid);                                             01386
      IF stid.stpsid # origin THEN                                  01387
         UNTIL getftl(stid) DO                                      01388
         stid ← getsuc(stid);                                       01389
      RETURN(stid);                                                 01390
      END.
                                                                    01391
   (getup)                                                          01392
      %Given an stid, this routine returns the stid of the source of
      the original stid%                                            01393
      %--------------%                                             01394
      PROCEDURE(stid);                                              01395
```

```
   IF stid.stpsid # origin THEN                                    01396
      stid ← getsuc(getail(stid));                                 01397
   RETURN(stid);                                                   01398
   END.
                                                                   01399
(gethed)                                                           01400
   %Given an stid, this routine returns the head of the current
   plex; if it is passed the origin statement, it returns the
   origin%                                                         01401
   %--------------%                                               01402
   PROCEDURE(stid);                                                01403
   IF stid.stpsid # origin THEN                                    01404
      stid ← getsub(getup(stid));                                  01405
   RETURN(stid);                                                   01406
   END.
                                                                   01407
(getprd)                                                           01408
   %Given an stid, this routine returns the predecessor; if the
   psid heads a plex, the stid itself is returned%                 01409
   %--------------%                                               01410
   PROCEDURE (stid);                                               01411
   LOCAL presid, %stid of predecessor%                             01412
      sucsid; %stid of successor of presid%                        01413
   IF getfnd(stid) THEN RETURN(stid);                              01414
   presid ← gethed(stid);                                          01415
   UNTIL (sucsid ← getsuc(presid)) = stid DO                       01416
      presid ← sucsid;                                             01417
   RETURN(presid);                                                 01418
   END.
                                                                   01419
(getend)                                                           01420
   %This procedure returns the end of the branch headed by the
   stid passed it.%                                                01421
   %--------------%                                               01422
   PROCEDURE(stid);                                                01423
   UNTIL (stid := getsub(stid)) = stid                             01424
   DO stid ← getail(stid);                                         01425
   RETURN(stid);                                                   01426
   END.
                                                                   01427
(getvnd) %find end of branch%                                      01481
   % This procedure finds the "end" of the branch begun by        01482
   the stid passed it; it expects, as a second argument, a         01483
   maximum level to be used in determining the end of the          01484
   branch.  If all statements in the branch fall beneath the       01485
   minimal level, it returns the the stid passed it. %             01486
   %--------------------%                                         01487
   PROC(stid, level);                                              01488
   LOCAL curlev, tmstid;                                           01489
   curlev ← getlev(stid);                                          01490
   UNTIL curlev >= level DO                                        01491
      BEGIN                                                        01492
      IF (stid := getsub(stid)) = stid THEN RETURN(stid);          01493
      stid ← getail(stid);                                         01494
      BUMP curlev;                                                 01495
      END;                                                         01496
```

```
      RETURN(stid);                                                    01497
      END.                                                             01498
                                                                       01499
   (getbck)                                                            01428
      %This procedure finds the back of the stid of the statemment
      passed it.  It does not observe viewspecs.%                      01429
      %-------------%                                                  01430
      PROCEDURE(stid);                                                 01431
      IF stid.stpsid = origin THEN RETURN(stid);                       01432
      IF getfhd(stid) THEN RETURN(getup(stid));                        01433
      stid ← getprd(stid);                                             01434
      IF (stid := getsub(stid)) = stid THEN RETURN(stid);              01435
      RETURN(getend(getail(stid)));                                    01436
      END.
                                                                       01437
   (getnxt)                                                            01438
      %This procedure finds the sequentially "next" statement, i.e.
      the substatement, successor, or successor of up, etc, of the
      stid passed as argument.  Ignores all viewspecs.%                01439
      %-----------------%                                              01440
      PROCEDURE (stid);                                                01441
      IF stid = endfil THEN err($"end of file exceeded");              01442
      IF (stid := getsub(stid)) = stid THEN                            01443
         % no substructure %                                           01444
         BEGIN                                                         01445
         LOOP                                                          01446
            BEGIN                                                      01447
            IF stid.stpsid = origin THEN RETURN (endfil);              01448
            IF getftl(stid) = 0 THEN EXIT; % not a tail %              01449
            stid ← getsuc(stid);                                       01450
            END;                                                       01451
         stid ← getsuc(stid);                                          01452
         END;                                                          01453
      IF stid.stpsid = origin THEN RETURN (endfil);                    01454
      RETURN (stid);                                                   01455
      END.                                                             01456
                                                                       01457
                                                                       01458

%Get's with Ring%
                                                                        0842
   (getsuc) %The stid for the successor field is returned.  If there
   is no successor, the stid of the up is returned.%
                                                                        0843
      PROCEDURE (stid);                                                0844
      LOCAL rngloc;                                                    0845
      lodrng(stid : rngloc);                                           0846
      IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                       0847
      stid.stpsid ← [rngloc].rsuc;                                     0848
      RETURN (stid) END.
                                                                       0849
   (getsub) %The STID in the sub field is returned.%
                                                                       0850
      PROCEDURE (stid);                                                0851
      LOCAL rngloc;                                                    0852
      lodrng(stid : rngloc);                                           0853
```

```
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0854
    stid.stpsid ← [rngloc].rsub;                                    0855
    RETURN (stid) END.
                                                                    0856

(getftl) %The logical value of the tail flag is returned.%
                                                                    0857
    PROCEDURE (stid);                                               0858
    LOCAL rngloc;                                                   0859
    lodrng(stid : rngloc);                                          0860
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");     0861
    RETURN ([rngloc].rtf) END.
                                                                    0862

(getfhd) %The logical value of the head flag is returned.%
                                                                    0863
    PROCEDURE (stid);                                               0864
    LOCAL rngloc;                                                   0865
    lodrng(stid : rngloc);                                          0866
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0867
    RETURN ([rngloc].rhf) END.
                                                                    0868

(getnmf) %The logical value of the name flag is returned.%
                                                                    0869
    PROCEDURE (stid);                                               0870
    LOCAL rngloc;                                                   0871
    lodrng(stid : rngloc);                                          0872
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0873
    RETURN ([rngloc].rnamef) END.
                                                                    0874

(getsdb) %The STDB in the specified ring element is returned.%
                                                                    0875
    PROCEDURE (stid);                                               0876
    LOCAL stdb, rngloc;                                             0877
    lodrng(stid : rngloc);                                          0878
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0879
    stdb ← 0;                                                       0880
    IF (stdb.stpsdb ← [rngloc].rsdb) = 0 THEN RETURN(0);            0881
    stdb.stfile ← stid.stfile;                                      0882
    RETURN (stdb) END.
                                                                    0883

(getnam) %The hash word for the statement name is returned.%
                                                                    0884
    PROCEDURE (stid);                                               0885
    LOCAL rngloc;                                                   0886
    lodrng(stid : rngloc);                                          0887
    IF [rngloc].rsid = 0 THEN err($"Bad statement identifier");
                                                                    0888
    RETURN ([rngloc].rnameh) END.
                                                                    0889

(getsid) %The statement identifier for the statement name is
returned.%
                                                                    0890
```

```
    PROCEDURE (stid);                                                   0891
    LOCAL rngloc;                                                       0892
    lodrng(stid : rngloc);                                              0893
    %this does NOT call err if sid=0 so that sid can be checked by
    other programs to test a ring element for validity%                 0894
    RETURN (/rngloc/.rsid) END.
                                                                        0895
%SDB Utility Routines%
                                                                        0464
  (lodsdb) %This routine loads an sdb into core.  The STDB is
  provided as argument.  Returns                                        0465
    (1) the CRPGAD index for the new page                               0466
    (2) the address of the sdb.%
                                                                        0467
    PROCEDURE (stdb);                                                   0468
    LOCAL                                                               0469
      wc,     %word count in block to the SDB%                          0470
      pgindx, %page index where loaded%                                 0471
      dt;     %pointer to DTBST entry for the page%                     0472
    REF dt;                                                             0473
    %wsd-meas%                                                          0474
      BUMP ldsdct;                                                      0475
    % this needs to be fast -- so we'll cheat and get the fields
    out of the stdb without using byte pointers.  the record
    definitions for stdb in utilty have a comment to the effect
    that we are doing this. %                                           0476
    r2 ← stdb;                                                          0477
    !HLRZ r1,r2;                                                        0478
    %the stfile field is right justified in the left halfword%          0479
    !HRRZ r2,r2;  %clear the left half%                                 0480
    !IDIVI r2,1B3; %split out the block number and the word count%
                                                                        0481
    wc ← r3;  % the dtbblk number is in r2 %                            0482
    &dt ← filehead/r1/ + $dtbst - $filhed + r2;                         0483
    %load if necessary%                                                 0484
    IF (pgindx ← dt.rfcore) = 0 THEN %must load it%                     0485
      pgindx ← lodrfb(r2+dtbbas, sdbtyp, r1);                           0486
    RETURN (pgindx, crpgad/pgindx/+wc) END.
                                                                        0487
  (fchsdb) %Shortcut for loading sdb given stid of statement.  See
  lodsdb for status when returns.%
                                                                        0488
    PROCEDURE (stid);                                                   0489
    LOCAL sdbloc;                                                       0490
    IF stid.stastr THEN err($"string treated as statement");            0493
    RETURN (lodsdb (getsdb(stid) : sdbloc), sdbloc) END.
                                                                        0494
  (fchtxt) %Called with stdb returns the number of the first
  character after the name of the statement%
                                                                        01505
    PROCEDURE (stdb);                                                   01506
    LOCAL sdbloc;                                                       01507
    lodsdb (stdb : sdbloc);                                             01508
    RETURN (/sdbloc/.sname) END.
                                                                        01509
  (newsdb) %get a new SDB.  Arguments are                               01510
```

```
(1) size of new SDB,                                                    01511
(2) file number from which to allocate.                                 01512
Returns the STDB of the SDB.%
                                                                        01513
PROCEDURE (room, fileno);                                               01514
LOCAL                                                                   01515
   choi2,  %second choice block%                                        01516
   choi3,  %third choice block%                                         01517
   least,  %number of used words in choi3%                              01518
   diff,   %excess space in garbage SDB%                                01519
   free,   %free space start%                                           01520
   sdbblk, %index in DTBST%                                             01521
   dt,     %pointer into DTBST%                                         01522
   pgindx, %page index for block in core%                               01523
   dtbsta, %address of DTBST for the file%                              01524
   dtbloc, %address of DTBL for the file%                               01525
   blkad,  %address of the block%                                       01526
   sdbpt,  %pointer to SDB's in the block%                              01527
   stdb,   %STDB for the new SDB%                                       01528
   fl;     %Address of file status table entry%                         01529
REF dt, fl;                                                             01530
dtbsta ← $dtbst + dtbloc ← filehead[fileno] - $filhed;                  01531
dtbloc ← dtbloc + $dtbl;                                                01532
stdb ← 0;                                                               01533
stdb.stfile ← fileno;                                                   01534
%check the block from which last allocated or freed                     01535
(number saved in DBLST)%                                                01536
&dt ← dtbsta + dblst;                                                   01537
IF NOT filepart[fileno] THEN %Make pc if neccessary%                    01538
   BEGIN                                                                01539
   &fl ← (fileno-1)*filstl + $filst;                                    01540
   IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT makepc(&fl,
   fileno, $lit) THEN                                                   01541
      BEGIN                                                             01542
      IF NOT fl.flbrws THEN lkun(&fl, $lit); %change user
      setable word to unlock%                                           01543
      dismes(2, $lit);                                                  01544
      GOTO STATE;                                                       01545
      END;                                                              01546
   END;                                                                 01547
IF dblst IN [0,dtbm) AND dt.rfexis AND                                  01548
   dt.rfcore AND dt.rffree + room <= blksiz THEN                        01549
      BEGIN %we won this time%                                          01550
      stdb.stblk ← dblst;                                               01551
      stdb.stwc  ← dt.rffree;                                           01552
      dt.rffree ← dt.rffree + room;                                     01553
      dt.rfused ← dt.rfused + room;                                     01554
      RETURN (stdb);                                                    01555
      END;                                                              01556
choi2 ← choi3 ← -1;                                                     01557
least ← blksiz;                                                         01558
sdbblk ← 0;                                                             01559
&dt ← dtbsta;                                                           01560
DO BEGIN                                                                01561
   IF dt.rfexis THEN %block already allocated%                          01562
      IF dt.rffree + room <= blksiz THEN                                01563
```

```
            %room in free space%                                    01564
            IF dt.rfcore THEN %loaded already%                      01565
               BEGIN                                                01566
               stdb.stblk ← dblst ← sdbblk;                         01567
               stdb.stwc ← dt.rffree;                               01568
               dt.rffree ← dt.rffree + room;                        01569
               dt.rfused ← dt.rfused + room;                        01570
               RETURN (stdb);                                       01571
               END                                                  01572
            ELSE choi2 ← sdbblk                                     01573
         ELSE %check if there is room if garbage collect%           01574
            %and page not frozen%                                   01575
            IF dt.rfused + room <= blksiz AND                       01576
               dt.rfused < least AND                                01577
               (dt.rfcore = 0 OR corpst[dt.rfcore].ctfroz = 0)
                                                                    01578
               THEN %new best choice%                               01579
               BEGIN                                                01580
               choi3 ← sdbblk;                                      01581
               least ← dt.rfused;                                   01582
               END;                                                 01583
      BUMP &dt;                                                     01584
      END                                                           01585
   UNTIL (sdbblk ← sdbblk+1) > [dtbloc];                            01586
   IF choi2 >= 0 THEN %enough free space but not in core%           01587
      BEGIN                                                         01588
      stdb.stblk ← dblst ← choi2;                                   01589
      &dt ← dtbsta + choi2;                                         01590
      stdb.stwc ← dt.rffree;                                        01591
      dt.rffree ← dt.rffree + room;                                 01592
      dt.rfused ← dt.rfused + room;                                 01593
      RETURN (stdb);                                                01594
      END;                                                          01595
   %if choi3 then block has enough room but may have to             01596
   garbage collect%                                                 01597
   IF choi3 >= 0 THEN                                               01598
      BEGIN                                                         01599
      stdb.stblk ← dblst ← choi3;                                   01600
      lodsdb(stdb : blkad);                                         01601
      &dt ← dtbsta + choi3;                                         01602
      free ← dt.rffree + blkad;                                     01603
      sdbpt ← blkad + fbhdl;                                        01604
      UNTIL sdbpt >= free DO                                        01605
         %try to find a garbage sdb that is big enough%             01606
         BEGIN                                                      01607
         IF [sdbpt].sgarb AND [sdbpt].slength >= room THEN          01608
            %will put the new SDB here%                             01609
            BEGIN                                                   01610
            stdb.stwc ← sdbpt - blkad;                              01611
            IF (diff ← [sdbpt].slength - room) > 0 THEN             01612
               BEGIN %make excess look like garbage sdb%            01613
               [sdbpt+room].sgarb ← TRUE;                           01614
               [sdbpt+room].slength ← diff;                         01615
                  %depends on sgarb, slength being in first        01616
                  word of header since diff may = 1%                01617
               END;                                                 01618
```

```
            dt.rfused ← dt.rfused + room;                                   01619
            RETURN (stdb);                                                  01620
            END;                                                            01621
         %go to the next SDB%                                               01622
         IF (sdbpt := sdbpt + [sdbpt].slength) >= sdbpt THEN                01623
            badfil(fileno);                                                 01624
         END;                                                               01625
      %must garbage collect the block%                                      01626
      gcol(choi3, fileno);                                                  01627
      stdb.stwc ← dt.rffree;                                                01628
      dt.rffree ← dt.rffree + room;                                         01629
      dt.rfused ← dt.rfused + room;                                         01630
      RETURN (stdb);                                                        01631
      END;                                                                  01632
   %have to allocate a block%                                               01633
   sdbblk ← 0;                                                              01634
   &dt ← dtbsta;                                                            01635
   DO BEGIN                                                                 01636
      IF NOT dt.rfexis THEN %will initialize this block%                    01637
         BEGIN                                                              01638
         pgindx ← lodrfb(sdbblk+dtbbas, niltyp, fileno);                    01639
         blkad ← crpgad[pgindx];                                            01640
         %finish initialization of block header%                            01641
         [blkad].fbind  ← sdbblk;                                           01642
         [blkad].fbtype ← sdbtyp;                                           01643
         %now set up the status table entry%                                01644
         dt.rfcore ← pgindx;                                                01645
         dt.rfexis ← TRUE;                                                  01646
         dt.rfused ← dt.rffree ← fbhdl+room;                                01647
         %update DTBL for the file%                                         01648
         IF sdbblk > [dtbloc] THEN [dtbloc] ← sdbblk;                       01649
         dblst ← sdbblk;                                                    01650
         stdb.stblk ← dblst;                                                01651
         stdb.stwc ← fbhdl;                                                 01652
         RETURN (stdb);                                                     01653
         END;                                                               01654
      BUMP &dt;                                                             01655
      END                                                                   01656
   UNTIL (sdbblk ← sdbblk+1) = dtbm;                                        01657
   %have exhausted data blocks%                                             01658
   err($"data storage full") END.                                           01659
```

(gcol) %called by newsdb to garbage collect the sdb block whose block number and file number are passed as arguments.%

```
                                                                           01660
   PROCEDURE (sdbblk, fileno);                                             01661
   LOCAL                                                                   01662
      pgindx,  %page index for core page holding the block%                01663
      blkad,  %address of the block%                                       01664
      blkfre, %address of free space in block%                             01665
      freewd, %pointer to first garbage block%                             01666
      sdbsiz, %size of SDB%                                                01667
      pt,     %pointer to SDB%                                             01668
      stdb,   %stdb used in fixing up linkages%                            01669
      stid,   %stid for fix up's%                                          01670
      dt;    %pointer to DTBST entry for the block%                        01671
```

```
    REF dt;                                                          01672
    stid ← 0;                                                        01673
    stdb ← 0;                                                        01674
    stdb.stfile ← fileno;                                            01675
    stid.stfile ← fileno;                                            01676
    stdb.stblk ← sdbblk;                                             01677
    &dt ← filehead[fileno] + $dtbst + sdbblk - $filhed;              01678
    pgindx ← lodsdb(stdb : blkad);                                   01679
    frzblk(pgindx, 1);                                               01680
    ON SIGNAL ELSE frzblk(pgindx, -1); %release frozen page%         01681
    blkfre ← blkad + dt.rffree; %start of free space%                01682
    freewd ← blkad + fbnd1; %first SDB%                              01683
    %move to first garbage sdb%                                      01684
    UNTIL [freewd].sgarb OR freewd >= blkfre DO                      01685
       freewd ← freewd + [freewd].slength;                           01686
    pt ← freewd;                                                     01687
    WHILE pt < blkfre DO                                             01688
       BEGIN                                                         01689
       IF (sdbsiz ← [pt].slength) = 0 THEN badfil(fileno);           01690
       IF NOT [pt].sgarb THEN                                        01691
          BEGIN                                                      01692
          mvbfbf(pt, freewd, sdbsiz); %move up the next good         01693
          sdb%                                                       01694
          %store new stdb%                                           01695
          stdb.stwc ← freewd-blkad;                                  01696
          stid.stpsid ← [freewd].spsid;                              01697
          stosdb(stid, stdb);                                        01698
          freewd ← freewd + sdbsiz;                                  01699
          END;                                                       01700
       pt ← pt + sdbsiz;                                             01701
       END;                                                          01702
    %finally update the status table%                                01703
    dt.rffree ← freewd - blkad;                                      01704
    IF dt.rffree # dt.rfused THEN badfil(fileno);                    01705
    frzblk(pgindx, -1);                                              01706
    RETURN END.                                                      01707

(fresdb) %If the statement associated with stid passed as argument
has an SDB then set it to garbage.%                                  01708

    PROCEDURE (stid);                                                01709
    LOCAL                                                            01710
       stdb,   %STDB of block to be freed%                           01711
       blkad,  %address of block containing the SDB%                 01712
       dt,     %pointer to DTBST entry for the block%                01713
       blkfre, %address of free space for the block%                 01714
       sdbblk, %address of SDB to be freed%                          01715
       sdbpt,  %pointer to SDB's in the block%                       01716
       wc,     %displacement in block of SDB to be freed%            01717
       blknxt, %another pointer to SDB's in the block%               01718
       drastic, %bad news flag, makepc blitzed the file%             01719
       fl;     %Address of file status table entry%                  01720
    REF dt, fl;                                                      01721
    stdb ← getsdb(stid);                                             01722
    IF stdb.stpsdb = 0 THEN RETURN;                                  01723
    wc ← stdb.stwc := 0;                                             01724
```

```
dblst ← stdb.stblk;                                                    01725
lodsdb(stdb :blkad);                                                   01726
sdbblk ← blkad + wc;                                                   01727
IF NOT filepart[stdb.stfile] THEN %Make pc if neccessary%              01728
   BEGIN                                                               01729
   &fl ← (stdb.stfile-1)*filstl + $filst;                              01730
   IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT makepc(&fl,
   stdb.stfile, $lit : drastic) THEN                                   01731
      BEGIN                                                            01732
      IF NOT fl.flbrws THEN lkun(&fl, $lit); %change user
      setable word to unlock%                                          01733
      dismes(2, $lit);                                                 01734
      IF drastic THEN BEGIN                                            01735
         pause(4000);                                                  01736
         nlsrst();                                                     01737
         END;                                                          01738
      GOTO STATE;                                                      01739
      END;                                                             01740
   END;                                                                01741
%pointer to the dtbst entry%                                           01742
&dt ← filehead[stdb.stfile] + $dtbst - $filhed +                       01743
stdb.stblk;                                                            01744
%record decrease in used count%                                        01745
dt.rfused ← dt.rfused - [sdbblk].slength;                              01746
%pointer to free space%                                                01747
blkfre ← dt.rffree + blkad;                                            01748
sdbpt ← blkad + fbhdl;                                                 01749
IF sdbblk > sdbpt THEN                                                 01750
   %the block to be freed is not the first%                            01751
   %merge the preceeding SDB if it is garbage%                         01752
   BEGIN                                                               01753
   %move sdbpt to the SDB in front of the one to be                    01754
   freed%                                                              01755
   LOOP CASE blknxt ← sdbpt + [sdbpt].slength OF                       01756
      = sdbblk : EXIT;                                                 01757
      <= sdbpt : badfil(stid.stfile);                                  01758
      ENDCASE sdbpt ← blknxt;                                          01759
   %if it is garbage, then merge%                                      01760
   IF [sdbpt].sgarb THEN                                               01761
      BEGIN                                                            01762
      [sdbpt].slength ← [sdbpt].slength +                              01763
      [sdbblk].slength;                                                01764
      sdbblk ← sdbpt;                                                  01765
      END;                                                             01766
   END;                                                                01767
[sdbblk].sgarb ← TRUE;                                                 01768
IF (blknxt ← [sdbblk].slength + sdbblk) = blkfre THEN                  01769
   %can add to free space%                                             01770
   dt.rffree ← sdbblk - blkad                                          01771
ELSE IF [blknxt].sgarb THEN                                            01772
   %merge the two%                                                     01773
   [sdbblk].slength ← [sdbblk].slength +                               01774
   [blknxt].slength;                                                   01775
RETURN;                                                                01776
END.
                                                                       01777
```

```
%SDB Copy%                                                              0763
                                                                        0764
   (copsdb) %Copies an SDB.  Arguments are                              0765
      (1) source STID,                                                  0766
      (2) destination STID.
      It is assumed that destination does NOT have an SDB currently
      associated with it.%
                                                                        0767
      PROCEDURE (source, dest);                                         0768
      LOCAL                                                             0769
         room,    %size of the SDB%                                     0770
         stdb,    %STDB for the new SDB%                                0771
         sdbnew, %address of the new SDB%                               0772
         rnl,     %pointer to destination ring element%                 0773
         rngb,    %ring block index for source%                         0774
         stdbold,    %stdb for the source%                              0775
         sdbold, %address of the source element%                        0776
         blknum, %block for new sdb%                                    0777
         oldnam; %name hash of the statement%                           0778
      REF rnl, sdbnew, sdbold;                                          0779
      rngb ← lodrng(source : &rnl);                                     0780
      frzblk(rngb, 1);                                                  0781
      ON SIGNAL ELSE frzblk(rngb, -1); %release frozen page%            0782
      stdbold ← 0;                                                      0783
      stdbold.stfile ← source.stfile;                                   0784
      stdbold.stpsdb ← rnl.rsdb;                                        0785
      blknum ← lodsdb(stdbold : &sdbold);                               0786
      frzblk (blknum, 1);                                               0787
      ON SIGNAL ELSE BEGIN                                              0788
         frzblk(rngb, -1);                                              0789
         frzblk(blknum, -1);                                            0790
         END;                                                           0791
      room ← sdbold.slength;                                            0792
      stdb ← newsdb (room, dest.stfile);                                0793
      lodsdb(stdb : &sdbnew);                                           0794
      mvbfbf (&sdbold, &sdbnew, room);                                  0795
      sdbnew.spsid ← dest.stpsid;                                       0796
      oldnam ← rnl.rnameh;                                              0797
      corpst[blknum].ctfroz ← corpst[blknum].ctfroz - 1;                0798
      corpst[rngb].ctfroz ← corpst[rngb].ctfroz - 1;                    0799
      ON SIGNAL ELSE; %disarm%                                          0800
      lodrng (dest : &rnl);                                             0801
      rnl.rnamef ← oldnam # 0;                                          0802
      rnl.rnameh ← oldnam;                                              0803
      rnl.rsdb ← stdb.stpsdb;                                           0804
      RETURN END.
                                                                        0805
%Get's with SDB%
                                                                        0806
   (getint) %Called with STID, returns initials of associated
   statement (left justified in word zero filled).%
                                                                        0807
      PROCEDURE (stid);                                                 0808
      LOCAL STRING string[5];                                           0809
      LOCAL sdbint, sdbloc;                                             0810
      fchsdb(stid : sdbloc);                                            0811
```

```
      *string* ← NULL;                                                    0812
      string[1] ← 0;                                                      0813
      IF (sdbint ← [sdbloc].sinit) .A 4B6 THEN %old syle inits%           0814
         string[1].oldint ← sdbint                                        0815
      ELSE trnsint(sdbint, $string);                                      0816
         %sdb initials in five-bit characters%                            0817
      RETURN(string[1]);                                                  0818
      END.
                                                                          0819
   (trnsint) PROCEDURE(initials, string);                                 0820
      %Given a word of 5-bit initials, this routine translates them
      and appends them to the string whose address is passed.%            0821
      REF string;                                                         0822
      LOCAL char, charno, count;                                          0823
      count ← 0;                                                          0824
      UNTIL (count ← count + 1) > 4 DO                                    0825
         BEGIN                                                            0826
         charno ← CASE count OF                                           0827
            =1: initials.cint1;                                           0828
            =2: initials.cint2;                                           0829
            =3: initials.cint3;                                           0830
            ENDCASE initials.cint4;                                       0831
         IF (char ← *trn5t7*[charno + 1]) = SP THEN EXIT LOOP             0832
         ELSE *string* ← *string*, char;                                  0833
         END;                                                             0834
      RETURN;                                                             0835
      END.
                                                                          0836
   (gettim) %Called with STID, returns time of last write for
   associated statement.%
                                                                          0837
      PROCEDURE (stid);                                                   0838
      LOCAL sdbloc;                                                       0839
      fchsdb(stid : sdbloc);                                              0840
      RETURN ([sdbloc].stime) END.
                                                                          0841
   (getstsize) %Called with STID, returns size in characters for
   associated statement.%
                                                                         01802
      PROCEDURE (stid);                                                  01803
      LOCAL sdbloc;                                                      01804
      fchsdb(stid : sdbloc);                                             01805
      RETURN ([sdbloc].schars) END.
                                                                         01806
%Pseudo interrupt for file pages%
                                                                          0896
   (wrtpsi) PROCEDURE;                                                    0897
      % save the accumulators %                                          0898
      svac1 ← r1; r1 ← $svacs; !BLT r1, svacse;                          0899
      s ← s + 400000400B;                                                0900
      wrpi();                                                            0901
      !HRLZI r1, svacs;                                                  0902
      !BLT r1, 17B;                                                      0903
      r1 ← svac1;                                                        0904
      !JSYS debrk;                                                       0905
      END.
```

```
                                                                    0906
(wrpi) % write pseudo interrupt routine %                           0907
   PROCEDURE;                                                       0908
   LOCAL trpw, trpd, tpga, tpgx, fl, ct, rf, topc, flg, drastic;
                                                                    0909
   REF fl, ct, rf;                                                  0910
   drastic ← FALSE;                                                01163
   % read the trap words %                                          0911
   r1 ← 4B5; !JSYS gtrpw;                                           0912
   trpw ← r1; trpd ← r2;                                            0913
   % make sure that really is a write trap %                        0914
   IF NOT trpw .A 4B6 THEN !JSYS haltf;                             0915
   tpga ← trpw .A 777B3;                                            0916
   FOR tpgx ← 1 UP UNTIL > rfpmax DO                                0917
      IF crpgad[tpgx] = tpga THEN                                   0918
         BEGIN                                                      0919
         % set up pointers %                                        0920
         &ct ← $corpst + tpgx;                                      0921
         &fl ← (ct.ctfile-1)*filstl + $filst;                       0922
         &rf ← crpgad[fl.flhead] + fbhdl - $filhed;                 0923
         &rf ← &rf + $rfbs + ct.ctpnum;                             0924
         IF rf.rfpart THEN % page from partial copy %               0925
            BEGIN                                                   0926
            IF fl.flpcread THEN                                    01133
               BEGIN                                               01134
               dismes(2, $"Cannot write on this file");            01135
               [levtab] ← $wpiabt;                                 01136
               RETURN;                                             01137
               END;                                                01138
            topc ← FALSE;                                           0927
            r2 ← 14B10;                                             0928
            END                                                     0929
         ELSE                                                       0930
            BEGIN % page from original file %                       0931
            % check if have a partial copy %                        0932
            IF NOT fl.flpart THEN % dont have one %                 0933
               IF (NOT fl.flbrws AND NOT lkfile(&fl)) OR NOT
               makepc(&fl, ct.ctfile, $lit : drastic) THEN          0934
                  BEGIN                                             0935
                  IF NOT fl.flbrws THEN lkun(&fl, $lit); %change
                  user setable word to unlock%                      0936
                  dismes(2, $lit);                                  0937
                  [levtab] ← IF drastic                             0938
                     THEN $supervisor                              01189
                     ELSE $wpiabt;                                 01190
                  RETURN END;                                       0939
            IF fl.flpcread THEN                                    01139
               BEGIN                                               01140
               dismes(2, $"Cannot write on this file");            01141
               [levtab] ← $wpiabt;                                 01142
               RETURN;                                             01143
               END;                                                01144
            topc ← TRUE;                                            0940
            rf.rfpart ← TRUE;                                       0941
            r2 ← 1004B8;                                            0942
            END;                                                    0943
```

```
            % change access %                                          0944
            r1.LH ← 4B5;                                               0945
            r1.RH ← tpga / 512;                                        0946
            !spacs(r1);  % 60B %                                       0947
            %must check if rm,w  or write%                            01182
            IF trpw .A 10B6 THEN  %read modify write%                 01183
               BEGIN                                                  01184
               (trpw.RH) ← (trpw.RH);   %must touch page to make
               private%                                               01186
               END                                                    01187
            ELSE (trpw.RH) ← trpd;  %do write for write only case!%
                                                                      01188
            % map to pc if needed %                                    0950
            IF topc THEN                                               0951
               BEGIN                                                   0952
               r1.RH ← tpga/512;                                       0953
               r1.LH ← 4B5;                                            0954
               r2.RH ← ct.ctpnum;                                      0955
               r2.LH ← f1.f1part;                                      0956
               !pmap(r1, r2, 14B10);                                   0958
               END;                                                    0960
            RETURN;                                                    0961
            END;                                                       0962
         % the write is not into a file page %                         0963
         r1 ← $"Illegal write at location ";                           0964
         !HRLI r1,7B2;                                                 0965
         !JSYS psout;                                                  0966
         r1 ← 101B; %primary output file%                              0967
         r2 ← (levllc - 1) .A 18M; % the location %                    0968
         r3 ← 8; %base 8%                                              0969
         !JSYS nout; % write the number %                              0970
         LOOP !JSYS haltf;                                             0971
         END.
                                                                       0972
      (wpiabt)PROC;                                                   01122
         GOTO STATE;                                                  01123
         END.                                                         01124
      (lkun) PROCEDURE (f1, astrng);                                   0973
         %Resets user setable word to unlock%                          0974
         LOCAL lokdir, lokinit;                                        0975
         REF f1, astrng;                                               0976
         r1 ← f1.florig;                                               0977
         r2 ← 100002AB; %read one word at 24B%                         0978
         r3 ← $r3;                                                     0979
         !JSYS gtfdb;                                                  0980
         lokdir ← r3.lkdirn;                                           0981
         lokinit ← r3.lkinit;                                          0982
         !JSYS gjinf;                                                  0983
         IF r1 = lokdir AND lokinit = cinit THEN                       0984
            IF NOT setfdb(0, 0, &f1) THEN                              0985
               %If user setable word has not been set back here, file is
               probably bad%                                           0986
               *astrng* ← *astrng*, "File is bad";                     0987
         RETURN;                                                       0988
         END.
                                                                       0989
```

```
(lockfile) PROCEDURE(fileno);                                          0990
   IF NOT lkfile(flntadr(fileno)) THEN                                 0991
      BEGIN                                                            0992
      dismes(2, $lit);                                                 0993
      RETURN(FALSE);                                                   0994
      END;                                                             0995
   RETURN(TRUE) END.
                                                                       0996
(lkfile) PROCEDURE(fl);                                                0997
   LOCAL                                                               0998
      dir, lkword, int, cnt, byt, bptr, bptr1, char, mask, value;
                                                                       01504
   LOCAL STRING intstr[5], dirname[30];                                 0999
   REF fl;                                                            01000
   !gtfdb( fl.florig, 1000024B, $r3);                                 01004
   lkword ← r3;                                                       01005
   IF NOT maywrt(0, &fl)
   OR NOT fl.flaccm .A accmask[lkword.acctyp] THEN                    01006
      BEGIN                                                           01007
      *lit* ← "No Write Access To ", *[fl.flastr]*;                   01008
      RETURN(FALSE);                                                  01009
      END;                                                            01010
   IF lkword.lkdirn NOT= 0 OR                                         01011
      lkword.lkinit NOT= 0 THEN %file is locked%                      01012
      BEGIN                                                           01013
      dir ← lkword.lkdirn;                                            01014
      int ← lkword.lkinit;                                            01015
      %See If user has locked it..if so, let it go%                   01016
      !JSYS gjinf;                                                    01017
      IF r1 = dir AND cinit = int THEN RETURN(TRUE);                  01018
      IF int .A 4B6 THEN %maybe locked with old style initials%
                                                                      01019
         BEGIN                                                        01020
         *intstr* ← NULL;                                             01021
         trnsint(cinit, $intstr);                                     01022
         IF intstr.L = 3 AND intstr[1].oldint = int THEN              01023
            RETURN(TRUE);                                             01024
         END;                                                         01025
      *lit* ← *[fl.flastr]*;                                          01026
      lockid($lit,  dir, int); %get lock ident string%                01027
      RETURN(FALSE);                                                  01028
      END;                                                            01029
   %Now get directory from PC name%                                   01030
      bptr ← chbptr(0) + fl.flpcst;                                   01031
      bptr1 ← chbptr(0) + $dirname;                                   01032
      WHILE  ↑bptr # '< DO NULL;                                      01033
      WHILE (char ←↑bptr) # '> DO ↑bptr1 ← char;                      01034
      ↑bptr1 ← 0;                                                     01035
      r1 ← 0;                                                         01036
      r2 ← chbmty + $dirname;                                         01037
      !JSYS stdir;                                                    01038
         GOTO lkfilerr;                                               01039
         GOTO lkfilerr;                                               01040
      dir ← r1.RH; %Directory Number%                                 01043
   mask ← 0;                                                          01046
   mask.lkinit ← mask.lkdirn ← 36M;                                   01047
```

```
    value.lkinit ← cinit;                                          01048
    value.lkdirn ← dir;                                            01049
    chnfdb(fl.florig, 24B, mask, value);                           01503
    RETURN(TRUE);                                                  01051
    (lkfilerr): err($"Illegal PC Name");                           01042
    END.
                                                                   01502
(lockid) PROCEDURE(astr, dirnum, initials);                        01052
    %append "Being Modified By dirnumnam (init)" to astr%          01053
    REF astr;                                                      01054
    LOCAL STRING intstr[5];                                        01055
    LOCAL cnt, byt;                                                01056
    *astr* ← *astr*, " Being Modified By ";                        01057
    IF dirnum AND SKIP !dirst(byt ← chbptr(astr.L) + &astr, dirnum)
    AND                                                            01060
        (cnt ← slngth(byt, rl)) + astr.L <= astr.M THEN            01061
            astr.L ← astr.L + cnt;                                 01062
    *intstr* ← NULL;                                               01063
    IF initials .A 4B6 THEN %old style initials%                   01064
        BEGIN                                                      01065
        intstr[1].oldint ← initials:                               01066
        intstr.L ← 3;                                              01067
        END                                                        01068
    ELSE trnsint(initials, $intstr);                               01069
    *astr* ← *astr*, SP, '(, *intstr*, ');                         01070
    RETURN END.
                                                                   01071
(makepc)PROC(fl, fileno, astr);                                    01072
    REF fl, astr;                                                  01073
    LOCAL jfn, i, flg;                                             01074
    %Create a PC for te indicated file, return any errors in astr.
                                                                   01075
    REturn TRUE if ok, FALSE otherwise%                            01076
    flg ← getgtjflg(write, chkpcf, 0);                             01077
    IF (fl.flpart ← rl ← sgtjfn(flg, fl.flpcst, &astr)) THEN BEGIN
                                                                   01145
        r2 ← 1000001B;                                             01147
        r3 ← $r3;                                                  01148
        !JSYS gtfdb;                                               01149
        IF SKIP !TLNE r3,600000B THEN  %exists already and not
        deleted%                                                   01150
            BEGIN                                                  01151
            thwfil(fileno);                                        01156
            delfil(fileno);                                        01152
            IF nlmode = fulldisplay THEN alldsp() ELSE tda.dacsp ←
            endfil;                                                01162
            *astr* ← "File Locking Conflict--Please Reload File";
                                                                   01154
            RETURN (FALSE, TRUE);                                  01161
            END;                                                   01155
        IF SKIP !TLNN r3,400000B THEN  %deleted%                   01157
            BEGIN %must undelete file%                             01158
            !HRLI r1,1; !HRLZI r2,400000B; r3 ← 0; !JSYS chfdb;    01160
            END;                                                   01159
        END;                                                       01146
    IF NOT fl.flpart OR                                            01078
```

```
   NOT sysopen(fl.flpart, readwrite, random, &astr) OR                  01079
   NOT sysclose(fl.flpart .V 4B11, &astr) THEN                          01080
      BEGIN                                                             01081
      IF (r1 ← fl.flpart) THEN                                          01082
         BEGIN %get rid of jfn%                                         01083
         IF SKIP !JSYS closf THEN NULL                                  01084
         ELSE                                                           01085
            %closf jsys failed (file may already be closed; set
            up r1 for r1jfn%                                            01086
            r1 ← fl.flpart;                                             01087
         IF SKIP !JSYS r1jfn THEN NULL;                                 01088
         END;                                                           01089
      fl.flpart ← 0;                                                    01090
      filepart[fileno] ← FALSE;                                         01091
      RETURN(FALSE, FALSE);                                             01092
      END;                                                              01093
filepart[fileno] ← TRUE;                                                01094
IF NOT sysopen(fl.flpart, readwrite, random, &astr) THEN                01095
   BEGIN %Wait fo 1 sec and try it again in case of a timing
   problem%                                                             01096
   r1 ← 1000;                                                           01097
   !JSYS dismS;                                                         01098
   IF NOT sysopen(fl.flpart, readwrite, random, &astr) THEN
                                                                        01099
      BEGIN                                                             01100
      IF (r1 ← fl.flpart) THEN %get rid of jfn%                         01101
         BEGIN                                                          01102
         IF SKIP !JSYS closf THEN NULL                                  01103
         ELSE                                                           01104
            %closf jsys failed (file may already be closed; set
            up r1 for r1jfn%                                            01105
            r1 ← fl.flpart;                                             01106
         IF SKIP !JSYS r1jfn THEN NULL;                                 01107
         END;                                                           01108
      fl.flpart ← 0;                                                    01109
      filepart[fileno] ← FALSE;                                         01110
      RETURN(FALSE, FALSE);                                             01111
      END;                                                              01112
   END;                                                                 01113
% change protection of pc to be same as file %                         01170
   !gtfdb(fl.florig, 1000004B, $r3); %protecton of original
   file%                                                                01171
   r1.LH ← 4B; %offset for protection word)%                            01172
   r1.RH ← fl.flpart; %jfn for partial copy%                            01173
   !chfdb(r1, 777777B, r3); %change righthalf bits only%                01174
%map header out to the pc%                                              01175
   r1.LH ← 485;                                                         01176
   r1.RH ← crpgad[fl.flhead] / 512;                                     01177
   r2.LH ← fl.flpart;                                                   01178
   r2.RH ← 0;                                                           01179
   r3 ← 14B10;                                                          01180
   !JSYS pmap;                                                          01181
%DELETE OLD PAGES EXCEPT FOR THE HEADER%                                02019
   jfn ← fl.flpart;                                                     02020
   FOR i ← 1 UP UNTIL >= 512 DO                                         02021
      BEGIN                                                             02022
```

```
            r1.LH ← jfn;  r1.RH ← i;                                    02023
            !rpacs(r1);                                                 02024
            IF r2 .A 1B10 THEN %page exists -- get rid of it%           02025
               BEGIN                                                    02026
               r2.LH ← jfn;  r2.RH ← i;                                 02027
               !pmap(-1, r2, 0);                                        02028
               END;                                                     02029
            END;                                                        02030
      RETURN(TRUE, FALSE) END.                                          01121

%File header location%                                                  01125
                                                                        01126
   (filhdr) PROCEDURE (fileno);
      %returns the address of the file header for the file whose
      number is passed%                                                 01127
      %---------------%                                                01128
      LOCAL fl;                                                         01129
      REF fl;                                                           01130
      &fl ← a(fileno-1)*filstl + $filst;                                01131
      RETURN (crpgad[fl.flhead] + fbhdl) END.                           01132

%test and set bounds of structure%                                      01779
                                                                        01780
   (grptst)
      %Given two stid's, this routine checks that they specify a
      legal group; it also returns them ordered (GRP1,GRP2). If the
      stid's do not form a legal group, an err($"illegal group") is
      issued.%                                                          01782
      %-------------%                                                  01783
      PROCEDURE(stid1, stid2);                                          01784
      LOCAL t1, %working stid1%                                         01785
         t2; %working stid2%                                            01786
      IF (stid1.stpsid # stid2.stpsid AND (stid1.stpsid = orgstid OR
      stid2.stpsid = orgstid))                                          01787
      OR stid1.stfile # stid2.stfile THEN err($"illegal group");
                                                                        01788
                                                                        01789
      t1 ← stid1; t2 ← stid2;                                           01790
      LOOP                                                              01791
         BEGIN %is stid2 on same level, and after, stid1%               01792
         IF t1 = stid2 THEN RETURN(stid1, stid2);                       01793
         IF getftl(t1) THEN LOOP                                        01794
            BEGIN %is stid1 on same level after stid2%                  01795
            IF t2 = stid1 THEN RETURN(stid2, stid1);
            IF getftl(t2) THEN err($"invalid group selection");
                                                                        01796
                                                                        01797
            t2 ← getsuc(t2);                                            01798
            END;                                                        01799
         t1 ← getsuc(t1);                                               01800
         END;
      END.                                                              01801
                                                                          04
```

FINISH of filmnp

FINIS

(MLK)FINTNLS (MLK)FINTNLS (MLK)FINTNLS (MLK)FINTNLS (MLK)FINTNLS
(MLK)FINTNLS (MLK)FINTNLS (MLK)FINTNLS (MLK)FINTNLS (MLK)FINTNLS

```
< NLS, FINTNLS.NLS;11, >, 16-OCT-74 12:42 DSM ;;;;
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
      ah = 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, qt, st, rentr;                           020
   DECLARE random = 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 r1,110init;  %initialize 110 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
         lhostn ← hostnumber(); %save logical host number%             080
         oprtty ← IF lhostn = utilhost THEN utiloprtty ELSE
         arcoprtty;                                                    081
         tenex ← tenexverno();                                         082
         IF tenex < 13200 THEN lpesc ← 33B;  % REMOVE AFTER 132 %
                                                                    01372
         string ← 0;                                                   083
   IF NOT continue THEN % 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 exquery THEN     % dcesnt 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
               !pbin;                                                  0131
               IF r1 = CA THEN EXIT LOOP;                              0132
               !pbout('?);                                             0133
               END;                                                    0134
            END;                                                       0135
         END;                                                          0136
      END;                                                             0137
% setup user-options or use system standards %                         086
   IF NOT uoget() THEN uoreset();                                      087
% now set user-options page to copy-on-write %                        088
   uoaccess($userdata / 1000B, racc .V cwacc);                         089
% terminal setup %                                                    090
   %get device%                                                        096
      getdev(console, nldevice);                                       0100
   % move to setdev when dcw does new tab stuff %                      0101
      IF nlmode = typewriter THEN setabs($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
   IF nlmode = typewriter THEN                                         0143
      BEGIN                                                            0144
      IF console < 45B THEN                                            0145
         BEGIN                                                         0146
```

```
                *nlsnam* ← "TNLS";                                      0147
                nlstyp ← tntyp;                                         0148
                END                                                     0149
             ELSE                                                       0150
                BEGIN                                                   0151
                *nlsnam* ← "NTNLS";                                     0152
                nlstyp ← nttyp;                                         0153
                END;                                                    0154
             END                                                        0155
          ELSE                                                          0156
             BEGIN                                                      0157
             IF console < 455 THEN                                      0158
                BEGIN                                                   0159
                *nlsnam* ← "DNLS";                                      0160
                nlstyp ← dntyp;                                         0161
                END                                                     0162
             ELSE *nlsnam* ← "NDNLS";                                   0163
             END;                                                       0164
          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
    % initialize display and work station
     or control character output control%                              0174
       initdis();                                                       0175
    % startup back end %                                               0176
       startback(0);                                                    0177
    % 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 ← fintadr(fileno);                                  0188
                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.flcrig, 1000248, $r3);                    0195
                   IF r3.ojdelf THEN %new mail%                         0196
                      BEGIN                                             0197
```

```
                    r2 ← 0; r2.cjdelf ← 1;                                  0198
                    chnfdb (f1.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
```

```
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
```

```
(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,i1Oinit;                                              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 1h7B; %Reset jsys-- cannot use symbol because of
```



```
      conflict%                                                     01382
       LOOP !haltf:                                                 01383
```

```
                                                                    01384
   END;                                                             01385
% high que so this stuff goes fast %                                0245
   setpriority(202B);                                               0246
% initialize character tables and terminal stuff%                   0247
   initch(devtiex);    % for query and general startup %            0248
   initbtbl();         % will be overlayed by useroptions data
   %                                                                0249
   feedbk ← deifdbk;                                                0250
   echon();                                                         0251
% check for loading page boundaries overflow %                      0252
   intmsf ← FALSE;  % no warning initially %                        0253
   IF bndchk THEN                                                   0254
      BEGIN                                                         0255
      IF $uopstart >= $userdata THEN                                0256
         BEGIN                                                      0257
         typeas($"

         ************************************
         HIGH SEQ LOADS IN USER-PROFILE PAGE
         ************************************
         ");                                                        0258
         intmsf ← TRUE;                                             0259
         END;                                                       0260
      IF ($enduop - $userdata) > 1000B THEN                         0261
         BEGIN                                                      0262
         typeas($"

         ***************************
         USER-PROFILE PAGE TOO LONG
         ***************************
         ");                                                        0263
         intmsf ← TRUE;                                             0264
         END;                                                       0265
      IF lowsegtop.LH >= symtbltop.RH THEN                          0266
         BEGIN                                                      0267
         typeas($"

         ***************************
```

```
    **************************
    CODE RUNS INTO SYMBOL TABLE
    **************************
    ");                                                          0268
    intmsf ← TRUE;                                               0269
    END                                                          0270
ELSE                                                             0271
    IF (lowsegtop.LH + 2000B) >= symtbltop.RH THEN               0272
       BEGIN                                                     0273
       typeas($"

       ********************************
       LESS THAN 2 PAGES FOR NEW SYMBOLS
       ********************************
       ");                                                       0274
```

```
                intmsf ← TRUE;                                          0275
                END;                                                    0276
            IF intmsf THEN *intmsg* ← EOL, "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 - $dspblk-1);             0285
        % initialize system default viewspecs %                         0286
          defvs1 ← 0;                                                   0287
          defvs2 ← 0;                                                   0288
          defvs1.vstrnc ← defvs1.vslev ← 63;                            0289
          defvs1.vsindf ← defvs1.vsnamf ← defvs1.vsdaft
          ←defvs1.vsnamf ← defvs1.vspagf ←  TRUE;                       0290
        % random stuff %                                                0291
          exquery ← FALSE;   % normal is not query %                   0292
        % ask if there is a startup message %                           0293
          LOOP                                                          0294
            BEGIN                                                       0295
            typeas($"startup message?? ");                              0296
            !pbin;                                                      0297
            CASE rl OF                                                  0298
              = 'y, = 'Y: %yes%                                         0299
                BEGIN                                                   0300
                typeas($"es.
                ");                                                     0301
                gtintmsg();                                             0302
                EXIT LOOP;                                              0303
                END;                                                    0304
              = 'n, = 'N: %no%                                          0305
                BEGIN                                                   0306
                typeas($"o.
                ");                                                     0307
                EXIT LOOP;                                              0308
                END;                                                    0309
              ENDCASE                                                   0310
                BEGIN                                                   0311
                typeas($" ? (y or n)
                ");                                                     0312
                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
%.....Initialization support routines.....%                             0324
  (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
         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 520000E, 0);              0339
         % MODIFY BOUNDS LATER %                                       0340
   RETURN;                                                             0341
   END.                                                                0342
(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
   !sevec(4B5, 7B6 .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
      IF nldevice = devlproc THEN da.damrow ← da.damrow - vinc;        0377
      da.damcol ← IF nlmode NOT= typewriter THEN                       0378
         ((taright-taleft)/hinc)*hinc ELSE colmax * hinc %columns%;
```

```
                                                                    0379
      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
(getdev) PROCEDURE(lineno, device); %get user's device%             0415
   % 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%                                                         0416
   %-----------------%                                             0417
   LOCAL devtype, i, char, count, code;                             0418
   !gttyp(lineno .V 4B5); %line number designator%                  0419
   devtype ← r2;                                                    0420
   IF continue AND devtype = nldevice THEN RETURN;                  0421
   IF devtype = devlproc THEN                                       0422
      IF tenex < 13200 THEN                                        01262
         BEGIN                                                      0423
         !rlpmd;                                                    0424
         i ← 0;                                                     0425
```

```
    IF r1 = 0 THEN %interrogate has not been acknowledged yet%
                                                                    0426
       FOR i ← 0 UP UNTIL >= 10 DO                                  0427
          BEGIN                                                     0428
          intter();                                                 0429
          !dismS(1000);                                             0430
          !rlpmd;                                                   0431
          IF r1 NOT= 0 THEN EXIT LOOP;                              0432
          END;                                                      0433
    IF i >= 10 THEN %interrogate has not been acknowledged yet%
                                                                    0434
       BEGIN                                                        0435
       typeas($"No Line-Processor data from RLPMD JSYS in
       GETDEV");                                                    0436
       LOOP !haltf;                                                 0437
       END;                                                         0438
    lpxmax ← r1.LH;                                                 0439
    lpymax ← r1.RH;                                                 0440
    lptype ← r2;                                                    0441
    lpbaudfactor ← r3;                                              0442
    lpdlpad ← r4;                                                   0443
    litcolumn[0] ← 1;                                               0444
    IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN             0445
       err($"GTJFN failed, GETDEV");                                0446
    dspjfn ← r1;                                                    0447
    IF NOT SKIP !openf(dspjfn, 100000385) THEN                      0448
       err($"OPENF failed, GETDEV");                                0449
    CASE lptype .A 4M OF                                            0450
       = hazeltine: putbackchar ← TRUE;                             0451
       ENDCASE putbackchar ← FALSE;                                 0452
    ldspjfn ← 0;  %no linked line-processors%                       0453
    END                                                             0454
ELSE                                                                01263
    BEGIN                                                           01264
    *altinp* ← NULL;                                                01302
    FOR i ← 0 UP UNTIL >= 10 DO                                     01268
       BEGIN                                                        01269
       % send and read big characters %                             01304
          !sfmd2(100B, 4B11 + 3B11 );                               01303
       intter();                                                    01270
       !dismS(1000);                                                01271
       LOOP                                                         01310
          IF NOT SKIP !sibe(100B) THEN                              01305
             BEGIN                                                  01306
             !pbin();                                               01307
             *altinp* ← *altinp*, (char ← r1);                      01352
             IF char # lpesc THEN                                   01309
                REPEAT LOOP;                                        01311
             !pbin();                                               01312
             *altinp* ← *altinp*, (count ← r1);                     01353
             IF count = lpesc THEN                                  01313
                REPEAT LOOP;                                        01314
             IF (count ← count-40B) <= 0 THEN REPEAT LOOP;          01318
             !pbin();                                               01321
             *altinp* ← *altinp*, (code ← r1);                      01354
             IF code # lrep THEN                                    01322
```

```
                    BEGIN                                                01335
                    WHILE (count ← count-1) DO                           01326
                       BEGIN                                             01327
                       !pbin();                                          01328
                       *altinp* ← *altinp*, r1;                          01329
                       END;                                              01331
                    REPEAT LOOP;                                         01336
                    END;                                                 01337
                 altinp.L ← altinp.L - 3;                                01338
                 !pbin();                                                01332
                 lpxmax ← r1 - 40B;                                      01333
                 !pbin();                                                01339
                 lpymax ← r1 - 40B;                                      01340
                 !pbin();                                                01349
                 lptype ← r1 - 40B;                                      01344
                 !pbin();                                                01350
                 lpdlpad ← r1 - 40B;                                     01345
                 !pbin();                                                01351
                 lpbaudfactor ← r1 - 40B;                                01346
                 EXIT LOOP 2;                                            01324
                 END                                                     01325
              ELSE REPEAT LOOP 2;                                        01306
           END;                                                          01274
        IF i >= 10 THEN %interrogate has not been acknowledged yet%
                                                                         01275
           BEGIN                                                         01276
           typeas($"No Line-Processor data from RLPMD JSYS in
           GETDEV");                                                     01277
           LOOP !naltf;                                                  01278
           END;                                                          01279
        litcolumn[0] ← 1;                                                01285
        !rfmod(100B);                                                    01355
        r2 ←                                                             01356
           r2 .A 740000777777B .V (lpymax * 2B8) .V (lpxmax * 1B6);
                                                                         01358
        !sfmod( 100B, r2);                                               01357
        r2 ← 4B10 + ((lpdlpad / lpbaudfactor) * 2B8)  % LF padding
        %                                                                01359
           + 4B6  % 2 input buffers %                                    01360
           + ((2 + (8/lpbaudfactor)) * 1B3)  % # ouptut buffers %
                                                                         01361
           + 6B11;  % sending and reading Big Chars %                    01362
        !sfmd2( 100B, r2 );                                              01363
        IF altinp.L THEN &rawchr ← $!paltgetchar;                        01364
        IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN              01286
           err($"GTJFN failed, GETDEV");                                 01287
        dspjfn ← r1;                                                     01288
        IF NOT SKIP !openf(dspjfn, 1000003B5) THEN                       01289
           err($"OPENF failed, GETDEV");                                 01290
        CASE lptype .A 4M OF                                             01291
           = hazeltine: putbackchar ← TRUE;                              01292
           ENDCASE putbackchar ← FALSE;                                  01293
        ldspjfn ← 0;  %no linked line-processors%                        01294
        END;                                                             01295
  IF devtype = imlac0 OR devtype = imlac1 THEN                            0455
     BEGIN                                                               0456
```

```
    IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN                    0457
       err($"GTJFN failed, GETDEV");                                       0458
    dspjfn ← r1;                                                           0459
    IF NOT SKIP !openf(dspjfn, 100000385) THEN                             0460
       err($"OPENF failed, GETDEV");                                       0461
    ldspjfn ← 0;  %no linked line-processors%                              0462
    END;                                                                   0463
 CASE device OF                                                            0464
    = devsri: %user wants DNLS%                                            0465
       CASE devtype OF                                                     0466
          = devsri, = imlac0, = imlac1, = devlproc: NULL;                  0467
          ENDCASE %not good%                                               0468
             BEGIN                                                         0469
             typeas($"not a display terminal");                            0470
             crlf();                                                       0471
             !JSYS haltf;                                                  0472
             END;                                                          0473
    = devtiex: %user wants TNLS%                                           0474
       CASE devtype OF                                                     0475
          = devsri, = imlac0, = imlac1, = devlproc:                        0476
             devtype ← devtiex;                                            0477
          ENDCASE;                                                         0478
    = offline:  %user wants DEX%                                           0479
       devtype ← offline;                                                  0480
    ENDCASE; %use devtype%                                                 0481
 CASE devtype OF                                                           0482
    =dev33, =dev35, =dev37, =devtiex, =devsri, =imlac0, =imlac1,
    =nettty, =offline, = devlproc:                                         0483
       %defined device type%                                               0484
    ENDCASE devtype ← devtiex; %undefined, use devtiex%                    0485
 setdev(devtype);                                                          0486
 RETURN;                                                                   0487
 END.                                                                      0488
                                                                           0489
(setdev) PROCEDURE (device);                                               0490
   %set up device dependent data for passed device%                        0491
   LOCAL store, i;                                                         0492
   CASE nldevice ← device OF                                               0493
      = devsri:     %our local displays%                                   0494
         BEGIN                                                             0495
         % normal char size: 1; hinc: 12, 14, 18, 20; vinc: 25, 30,
         35, 40; grid: x: 0 to 1023 by y: 256 to 1023;%                    0496
         nlmode ← fulldisplay;                                             0497
         tmarg ← lmarg ← 0;                                                0498
         rmarg ← 1016*256;                                                 0499
         bmarg ← 1023*256;                                                 0500
         tacsize ← cflcsize ← namcsize ← subcsize ← litcsize ←
         msgcsize ← 1;                                                     0501
         ltcsize ← vscsize ← 0;                                            0502
         tahinc  ← cflhinc ← namhinc ← subhinc ← lithinc ← msghinc ←
         14*256;                                                           0503
         lthinc ← vshinc ← 12*256;                                         0504
         tavinc  ← cflvinc ← namvinc ← subvinc ← litvinc ← msgvinc ←
         ltvinc ← vsvinc ← 30*256;                                         0505
         intdspda(tmarg + 15*tavinc, (bmarg/tavinc)*tavinc, lmarg,
         (rmarg/tahinc)*tahinc);                                           0506
```

```
      %initialize display areas for display terminal%          0507
    END;                                                       0508
= 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
```

```
        &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
        nlmode ← typewriter;                                               0554
        intdspda(0, pgsize, tpcffset, 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
    littop ← 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 ← tar;                                                   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
      littop ← tatop - 2*tavinc;                                       0643
      litbottom ← tabottom;                                            0644
      litleft ← taleft;                                                0645
      litright ← taright;                                              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,
```

```
   netty%                                                                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[21B] ← $ascbst; %↑Q%                                       0684
      trnslo[$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
      r2 ← IF nlmode = typewriter THEN ttycoc ELSE dpycoc;             0724
      r3 ← 1B3;                                                        0725
      !JSYS sfcoc;                                                     0726
      END;                                                             0727
   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)%                                                  01371
         IF tenex < 13200 THEN !sccim(1)                              01367
         ELSE                                                         01368
            BEGIN                                                     01369
            !rfmd2(100B);                                              0730
            !sfmd2( 100B, r2 .V 4B11);                               01366
            END;                                                     01370
      CASE nldevice OF %turn tty simulation off%                       0731
         = devlproc: NULL;                                             0732
         = imlac0, = imlac1:                                           0733
            BEGIN                                                      0734
            *send* ← begmsg, 1+remfudge, remtsf;                       0735
            !scout(dspjfn, chbmty+$send, -send.L);                     0736
            END;                                                       0737
          ENDCASE %tasker%                                             0738
            IF NOT SKIP !uasqd(14B10) THEN !JSYS haltf;                0739
      END;                                                             0740
   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, =devsri, =imlac0, =imlac1:         0746
            IF NOT autostrt THEN                                       0747
               BEGIN                                                   0748
               r1 ← 100B;                                              0749
               !JSYS rfmod;                                            0750
               r2 ← r2 .A 777777777717B .V 4B10;                       0751
               !JSYS sfmod;                                            0752
               END;                                                    0753
         ENDCASE;                                                      0754
      END;                                                             0755
   RETURN END.                                                         0756
                                                                       0757
(setdis) PROCEDURE; %set up display areas%
   %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
         !scout(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, $crlarw,
      randm);                                                   0788
      [cflda.dalsrt].rtxl ← [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.datcp, 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 ← $dpyarea) + dal*dacnt;                     0907
         UNTIL &da >= end DO                                     0908
            BEGIN                                                0909
            IF da.daexis AND NOT da.dasuppress THEN              0910
               <NLS, DSPGEN, alocda>(&da);                       0911
               dassnbit(da.dalsidb, -1, lsbtsize);  %deassign all
               bits%                                             0912
            &da ← &da + dal;                                     0913
            END;                                                 0914
      END;                                                       0915
   echoff();                                                     0916
   RETURN;                                                       0917
   END.                                                          0918
                                                                 0919
                                                                 0920
(ttywindow) PROCEDURE (da);  %make a da a tty window%            0921
   LOCAL STRING send[20];                                        0922
   REF da;                                                       0923
   CASE nldevice OF                                              0924
      = deviproc:                                                0925
         lpttywindow(da.datop, da.dabottom-da.davinc);           0926
      = imlac0, = imlac1: %send out char string%                 0927
         BEGIN                                                   0928
         IF NOT da.dahandle THEN err($"Zero dahandle in ttywindow");
                                                                 0929
         *send* ← begmsg, 4+remfudge, echda, 508, da.dahandle.daidr1
         + remfudge, da.dahandle.daidr2 + remfudge;              0930
         !sout(dspjfn, chbmty+$send, -send.L);                   0931
         END;                                                    0932
      ENDCASE                                                    0933
         %issue uasqd jsys to make the seqda a copy of tty-sim da%
                                                                 0934
            IF NOT SKIP !uasqd(400000001B3 .V da.dahandle) THEN  0935
               err($"UASQD failed, setdis");                     0936
   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.daseq ← 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) + &strl;                            0971
         ls.rtbpe ← chbptr(strl.L) + &strl;                           0972
         IF &str2 THEN &strl ← &str2;                                 0973
         ls.rtcsize ← base.icsize;                                    0974
         ls.rtfont ← base.ifont;                                      0975
         ls.rthinc ← base.ihinc;                                      0976
         ls.rtxl ← 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);                                                     0986
   % 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. %                          0987
    LOCAL i, jfn, chain, bp, id[2], wheelf, pcap;                     0988
   LOCAL STRING errorstr[200];                                        0989
   wheelf ← IF (pcap ← enablw()) = -1 THEN 0 ELSE 1;                  0990
   disablw(pcap:=-1);                                                 0991
   jfn ← 0;                                                           0992
   ON SIGNAL ELSE                                                     0993
      BEGIN                                                           0994
      ON SIGNAL ELSE;                                                 0995
```

```
    IF jfn THEN sysclose(jfn);                                          0996
    jfn ← 0;                                                            0997
    END;                                                                0998
  %read id from file or get from user%                                  0999
    IF NOT (jfn ← sgtjfn(1B5, $"<SYSTEM>GROUP.INDEX;1",
    $errorstr)) THEN                                                    01000
      err($"File associating login directories with NLS idents is
      bad.  Please report to ARC personnel.");                          01001
    IF NOT sysopen(jfn, rthawed, bintyp, $errorstr) THEN                01002
      err($"File associating login directories with NLS idents is
      bad.  Please report to ARC personnel.");                          01003
    %get ident from group.index file or user%                           01004
      IF NOT SKIP !sfptr(jfn, user*4) THEN                              01005
        err($"File associating login directories with NLS idents
        is bad.  Please report to ARC personnel.");                     01006
      !bin(jfn); %get user directory number and group number%
                                                                        01007
      IF r2.LH NOT= user THEN                                           01008
        err($"File associating login directories with NLS idents
        is bad.  Please report to ARC personnel.");                     01009
      !bin(jfn); %get first word of ident or link to chain of
      idents%                                                           01010
      IF r2 = 0 THEN                                                    01011
        BEGIN                                                           01012
        IF NOT wheelf THEN err($"no idents associated with this
        login directory");                                              01013
        idfrmuser ($initsr);                                            01014
        END                                                             01015
      ELSE IF r2.LH = 18M THEN %chain of idents%                        01016
        BEGIN                                                           01017
        chain ← r2.RH;                                                  01018
        %get ident from user%                                           01019
          idfrmuser ($initsr);                                          01020
        LOOP                                                            01021
          BEGIN                                                         01022
          IF NOT SKIP !sfptr(jfn, chain) THEN                           01023
            err($"cant set position in file associating login
            directories an idents.  Please report to ARC
            personnel.");                                               01024
          !bin(jfn); %first word of ident%                              01025
          id ← r2;                                                      01026
          !bin(jfn); %second word of ident%                             01027
          id[1] ← r2;                                                   01028
          IF initsr[1] = id AND initsr[2] = id[1] THEN                  01029
            EXIT LOOP; %found a match, everything is ok%                01030
          !bin(jfn); %follow chain to next entry%                       01031
          IF (chain ← r2.RH) = 0 THEN %end of chain%                    01032
            BEGIN                                                       01033
            IF NOT wheelf THEN err($"ident not in list
            associated with this login directory");                     01034
            typeas($"using ident not associated with this
            login directory");                                          01035
            EXIT LOOP;                                                  01036
            END;                                                        01037
          END;                                                          01038
        END                                                             01039
```

```
      ELSE %get the ident%                                              01040
         BEGIN                                                          01041
         initsr[1] ← r2;                                                01042
         !bin(jfn);                                                     01043
         initsr[2] ← r2;                                                01044
         bp ← ch8mty + $initsr;                                         01045
         FOR i ← 1 UP UNTIL > 10 DO                                     01046
            IF ↑bp = 0 THEN EXIT LOOP;                                  01047
         initsr.L ← i-1;                                                01048
         END;                                                           01049
      CASE initsr.L OF                                                  01050
         > 4:                                                           01051
         err($"ident too long for use with NLS (max length is 4
         characters)");                                                 01052
         < 1: err($"Illegal ident.  Please report to ARC system
         personnel.");                                                  01053
         ENDCASE;                                                       01054
   sysclose(jfn);                                                       01055
   jfn ← 0;                                                             01056
   %now put initials into cinit, packed into 5-bit%                     01057
      cinit ← setcinit($initsr);                                        01058
   RETURN;                                                              01059
   END.
                                                                        01060
(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($"Illegal character in id");                       01070
      CASE count OF                                                     01071
         =1: val.cint1 ← char;                                          01072
         =2: val.cint2 ← char;                                          01073
         =3: val.cint3 ← char;                                          01074
         ENDCASE val.cint4 ← char;                                      01075
      END;                                                              01076
   RETURN (val);                                                        01077
   END.
                                                                        01078
(idfrmuser) PROCEDURE (str);  %collect ident from user in tty mode%
                                                                        01079
   LOCAL char;                                                          01080
   REF str;                                                             01081
   LOOP                                                                 01082
      BEGIN                                                             01083
      typeas($"
      Ident = ");                                                       01084
      *str* ← NULL;                                                     01085
      echon();                                                          01086
      LOOP                                                              01087
         BEGIN                                                          01088
         CASE (char ← rawchr()) OF                                      01089
```

```
            = '?: %help him%                                                01090
               BEGIN                                                        01091
               typeas($"
               If you have already been assigned a unique identifier
               (called an IDENT)");                                         01092
               typeas($" by the ARC Identification system (and you
               remember it), ");                                            01093
               typeas($"then please type it followed by a Carriage
               Return.");                                                   01094
               typeas($"  Otherwise, just type the ESC (ALTMODE) key
               on your terminal and contact ARC.
               ");                                                          01095
               REPEAT LOOP 2;                                               01096
               END;                                                         01097
            IN ['a, 'z]: char ← char - 40B;                                 01098
            IN ['A, 'Z], IN ['2, '6]: NULL;                                 01099
            = SP, =BW:  REPEAT LOOP 2;                                      01100
            =CA, =EOL, =CR: EXIT LOOP;                                      01101
            = $ascalt: !haltf;                                              01102
            =BC:                                                            01103
               BEGIN                                                        01104
               IF str.L THEN                                                01105
                  BEGIN                                                     01106
                  typecn('/);                                               01107
                  typech(*str*[str.L]);                                     01108
                  str.L ← str.L - 1;                                        01109
                  END;                                                      01110
               REPEAT CASE;                                                 01111
               END;                                                         01112
            ENDCASE                                                         01113
               BEGIN                                                        01114
               typeas($" ? ");                                              01115
               REPEAT CASE;                                                 01116
               END;                                                         01117
         *str* ← *str*, char;                                               01118
         END;                                                               01119
      IF str.L THEN EXIT LOOP;                                              01120
      END;                                                                  01121
   RETURN;                                                                  01122
   END.                                                                     01123
                                                                            01124
(setinterrupts)PROC;                                                        01125
   %set up rubout, command delete, and page write interrupt%                01126
      %IF YOU CHANGE THIS CODE, SEE ALSO (utilty, trapcc) AND
      (utilty, notrapcc)%                                                   01127
      chntab[1] ← $stopline .V 1B6; % ↑P %                                  01128
      %note: the ↑C dispatch is altered/restored by the journal%
                                                                            01129
      chntab[2] ← $ctrlc .V 1B6; %handle ↑C to do imlac stuff%              01130
      chntab[3] ← $rubout .V 1B6; % treat ↑O like a rubout %                01131
      chntab[4] ← $msgpsi .V 1B6; % for message fork to use %               01132
      chntab[9] ← $stkovr .V 1B6;                                           01133
      chntab[11] ← $iodaterr .V 1B6;                                        01134
      chntab[15] ← $ilspsi .V 1B6;  % illegal instruction %                 01135
      chntab[17] ← $wrtpsi .V 1B6;                                          01136
      chntab[24] ← $gotohelp .V 1B6;                                        01137
```

```
      r1 ← 4B5;                                                          01138
      r2 ← $chntab;                                                      01139
      !HRLI r2, levtab;                                                  01140
      !JSYS sir;                                                         01141
   % special stuff for control-c (↑C)%                                   01142
      r1 ← 4B5;                                                          01143
      !JSYS rpcap;                                                       01144
      r3 ← r3 .V 4B11;                                                   01145
      !JSYS epcap; %Permits process to assign ↑C for pseudo
      interrupt%                                                         01146
   %enable interrupts%                                                   01147
      r1 ← 4B5;                                                          01148
      !JSYS eir;                                                         01149
   %activate io data error and write pseudo interrupt%                   01150
      r1 ← 4B5;                                                          01151
      r2 ← 105B6; %channel 11, 15, 17%                                   01152
      !JSYS aic;                                                         01153
   %activate rubout, etc.%                                               01154
      r1 ← 200000018; %↑P%                                               01155
      !JSYS ati;                                                         01156
      r1 ← 170000038; %↑C%                                               01157
      !JSYS ati;                                                         01158
      r1 ← 210000308; %↑Q%                                               01159
      !JSYS ati;                                                         01160
      IF nldevice = imlac0 OR nldevice = imlac1 THEN                     01161
         BEGIN                                                           01162
         r1 ← 3000002B; %↑C%                                             01163
         !JSYS ati;                                                      01164
         r2 ← 3h0000000h000B; %activate rubout, ↑P, ↑Q, ↑C, ↑O%          01165
         END                                                             01166
      ELSE  r2 ← 2h0000000h000B; %activate ↑P, ↑Q and ↑O%                 01167
      r1 ← 4B5;                                                          01168
      !JSYS aic;                                                         01169
   RETURN;                                                               01170
   END.                                                                  01171

(gtintmsg) PROCEDURE;                                                    01172
   typeas($"  Message = ");                                              01173
   LOOP                                                                  01174
      BEGIN                                                              01175
      !pbin;                                                             01176
      IF r1 = CA THEN EXIT;                                              01177
      *intmsg* ← *intmsg*, r1;                                           01178
      END;                                                               01179
   IF intmsg.L THEN                                                      01180
      BEGIN                                                              01181
      *intmsg* ← *intmsg*, 0;                                            01182
      intmsf ← TRUE;                                                     01183
      END;                                                               01184
   RETURN(intmsg.L);                                                     01185
   END.                                                                  01186

(setabs)    % new bit table version of settabs %                         01187
   PROCEDURE                                                             01188
      (tabtbl);     % address of tab table %                            01189
   LOCAL twrd1, twrd2, twrd3;                                            01190
```

```
    REF tabtbl;                                                          01191
       !gjinf();                % get user info %                        01192
       IF r4 = -1 THEN RETURN;        % NOP if detached %               01193
    % convert NLS internal form to TENEX form %                          01194
       r1 ← tabtbl;                                                      01195
       r2 ← tabtbl[1];                                                   01196
       !SETCA 1,0;              %NLS bits are inverted%                  01197
       !SETCA 2,0;                                                       01198
       twrd1 ← r1;                                                       01201
       twrd2 ← r2;                                                       01202
       r2 ← tabtbl[2];                                                   01204
       !SETCA 2,0;                                                       01206
       twrd3 ← r2;                                                       01208
    % set tabs unless monitor words are the same %                      01209
       !gtabs(18M);                                                      01210
       IF twrd1 = r2 AND twrd2 = r3 AND twrd3 = r4 THEN RETURN           01211
       ELSE !stabs(18M, twrd1, twrd2, twrd3);                            01212
    RETURN;                                                              01213
    END.                                                                 01214
                                                                         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;                                                   01227
         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
```

```
< NLS, FINTNLS.NLS;11, >, 16-OCT-74 12:42 DSM ;;;;
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
      al = 12, %working register%                                       09
      ah = 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, qt, 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 r1,110init;  %initialize 110 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
         (strhlt):                                                     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
         lhostn ← hostnumber(); %save logical host number%             080
         oprtty ← IF lhostn = utilhost THEN utiloprtty ELSE
         arcoprtty;                                                    081
         tenex ← tenexverno();                                         082
         IF tenex < 13200 THEN lpesc ← 33B;  % REMOVE AFTER 132 %
                                                                    01372
         string ← 0;                                                   083
   IF NOT continue THEN % 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 exquery THEN     % dcesnt 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
         !pscut(chbmty + $intmsg);                                       0124
         crlf();                                                         0125
         IF nlmode = fulldisplay THEN                                    0126
            BEGIN                                                        0127
            typeas($intca);                                              0128
            LOOP                                                         0129
               BEGIN                                                     0130
               !pbin;                                                    0131
               IF r1 = CA THEN EXIT LOOP;                                0132
               !pbout('?);                                               0133
               END;                                                      0134
            END;                                                         0135
         END;                                                            0136
      END;                                                               0137
% setup user-options or use system standards %                           086
   IF NOT uoget() THEN uoreset();                                        087
% now set user-options page to copy-on-write %                          088
   uoaccess($userdata / 1000B, racc .V cwacc);                           089
% terminal setup %                                                      090
   %get device%                                                          096
      getdev(console, nldevice);                                         0100
   % move to setdev when dcw does new tab stuff %                       0101
      IF nlmode = typewriter THEN setabs($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
   IF nlmode = typewriter THEN                                           0143
      BEGIN                                                              0144
      IF console < 45B THEN                                              0145
         BEGIN                                                           0146
```

```
                *nlsnam* ← "TNLS";                                      0147
                nlstyp ← tntyp;                                         0148
                END                                                     0149
             ELSE                                                       0150
                BEGIN                                                   0151
                *nlsnam* ← "NTNLS";                                     0152
                nlstyp ← nttyp;                                         0153
                END;                                                    0154
             END                                                        0155
          ELSE                                                          0156
             BEGIN                                                      0157
             IF console < 45B THEN                                      0158
                BEGIN                                                   0159
                *nlsnam* ← "DNLS";                                      0160
                nlstyp ← dntyp;                                         0161
                END                                                     0162
             ELSE *nlsnam* ← "NDNLS";                                   0163
             END;                                                       0164
          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());                                     0175
             % initialize fields to standard values for a file window
             %                                                          0172
       END;                                                             0173
    % initialize display and work station
     or control character output control%                              0174
       initdis();                                                       0175
    % startup back end %                                               0176
       startback(0);                                                    0177
    % 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
                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, symtoltop = 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 r1,110init;                                                  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
% high que so this stuff goes fast %                             0245
    setpriority(202B);                                           0246
% 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 >= $userdata THEN                            0256
          BEGIN                                                  0257
          typeas($"

          ************************************
          HIGH SEQ LOADS IN USER-PROFILE PAGE
          ************************************
          ");                                                    0258
          intmsf ← TRUE;                                         0259
          END;                                                   0260
       IF ($enduop - $userdata) > 1000B THEN                     0261
          BEGIN                                                  0262
          typeas($"

          ****************************
          USER-PROFILE PAGE TOO LONG
          ****************************
          ");                                                    0263
          intmsf ← TRUE;                                         0264
          END;                                                   0265
       IF lowsegtop.LH >= symtbltop.RH THEN                      0266
          BEGIN                                                  0267
          typeas($"

          ****************************
          CODE RUNS INTO SYMBOL TABLE
          ****************************
          ");                                                    0268
          intmsf ← TRUE;                                         0269
          END                                                    0270
       ELSE                                                      0271
          IF (lowsegtop.LH + 2000B) >= symtbltop.RH THEN         0272
             BEGIN                                               0273
             typeas($"

             ************************************
             LESS THAN 2 PAGES FOR NEW SYMBOLS
             ************************************
             ");                                                 0274
```

```
                intmsf ← TRUE;                                         0275
                END;                                                   0276
            IF intmsf THEN *intmsg* ← EOL, "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 - $dspblk-1);            0285
        % initialize system default viewspecs %                        0286
          defvs1 ← 0;                                                  0287
          defvs2 ← 0;                                                  0288
          defvs1.vstrnc ← defvs1.vslev ← 63;                           0289
          defvs1.vsindf ← defvs1.vsnamf ← defvs1.vsdaft
          ←defvs1.vsnamf ← defvs1.vspagf ←  TRUE;                      0290
        % random stuff %                                               0291
          exquery ← FALSE;    % normal is not query %                  0292
        % ask if there is a startup message %                          0293
          LOOP                                                         0294
            BEGIN                                                      0295
            typeas($"startup message?? ");                             0296
            !pbin;                                                     0297
            CASE r1 OF                                                 0298
              = 'y, = 'Y: %yes%                                        0299
                BEGIN                                                  0300
                typeas($"es.
                ");                                                    0301
                gtintmsg();                                            0302
                EXIT LOOP;                                             0303
                END;                                                   0304
              = 'n, = 'N: %no%                                         0305
                BEGIN                                                  0306
                typeas($"o.
                ");                                                    0307
                EXIT LOOP;                                             0308
                END;                                                   0309
              ENDCASE                                                  0310
                BEGIN                                                  0311
                typeas($" ? (y or n)
                ");                                                    0312
                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
%.....Initialization support routines.....%                            0324
  (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
         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
(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
   !sevec(4B5, 7B6 .V $entvec); %set entry vector%                     0350
   RETURN;                                                             0351
   END.                                                                0352
(intdaf1) %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
   %displayarea 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
      IF nldevice = devlproc THEN da.damrow ← da.damrow - vinc;        0377
      da.damcol ← IF nlmode NOT= typewriter THEN                       0378
         ((taright-taleft)/hinc)*hinc ELSE colmax * hinc %columns%;
```

```
                                                                    0379
      da.dahinc ← hinc;                                             0380
      da.davinc ← vinc;                                             0381
      da.danoni ← hinc;                                             0382
      da.dasgni ← 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
(getdev) PROCEDURE(lineno, device); %get user's device%             0415
   % 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%                                                         0416
   %-----------------%                                             0417
   LOCAL devtype, i, char, count, code;                             0418
   !gttyp(lineno .V #35); %line number designator%                  0419
   devtype ← r2;                                                    0420
   IF continue AND devtype = nldevice THEN RETURN;                  0421
   IF devtype = deviproc THEN                                       0422
      IF tenex < 13200 THEN                                        01262
         BEGIN                                                      0423
         !rlpmd;                                                    0424
         i ← 0;                                                     0425
```

```
    IF r1 = 0 THEN %interrogate has not been acknowldged yet%          0426
       FOR i ← 0 UP UNTIL >= 10 DO                                     0427
          BEGIN                                                        0428
          intter();                                                    0429
          !dismS(1000);                                                0430
          !rlpmd;                                                      0431
          IF r1 NOT= 0 THEN EXIT LOOP;                                 0432
          END;                                                         0433
    IF i >= 10 THEN %interrogate has not been acknowldged yet%         0434
       BEGIN                                                           0435
       typeas($"No Line-Processor data from RLPMD JSYS in
       GETDEV");                                                       0436
       LOOP !haltf;                                                    0437
       END;                                                            0438
    lpxmax ← r1.LH;                                                    0439
    lpymax ← r1.RH;                                                    0440
    lptype ← r2;                                                       0441
    lpbaudfactor ← r3;                                                 0442
    lpdlpad ← r4;                                                      0443
    litcolumn[0] ← 1;                                                  0444
    IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN                0445
       err($"GTJFN failed, GETDEV");                                   0446
    dspjfn ← r1;                                                       0447
    IF NOT SKIP !openf(dspjfn, 1000003B5) THEN                         0448
       err($"OPENF failed, GETDEV");                                   0449
    CASE lptype .A 4M OF                                               0450
       = hazeltine: putbackchar ← TRUE;                                0451
       ENDCASE putbackchar ← FALSE;                                    0452
    ldspjfn ← 0;  %no linked line-processors%                          0453
    END                                                                0454
ELSE                                                                   01263
    BEGIN                                                              01264
    *altinp* ← NULL;                                                   01302
    FOR i ← 0 UP UNTIL >= 10 DO                                        01268
       BEGIN                                                           01269
       % send and read big characters %                                01304
          !sfmd2(100B, 4B11 + 3B11 );                                  0130
       intter();                                                       01270
       !dismS(1000);                                                   01271
       LOOP                                                            01310
          IF NOT SKIP !sibe(100B) THEN                                 01305
             BEGIN                                                     01306
             !pbin();                                                  01307
             *altinp* ← *altinp*, (char ← r1);                         01352
             IF char # lpesc THEN                                      01309
                REPEAT LOOP;                                           01311
             !pbin();                                                  01312
             *altinp* ← *altinp*, (count ← r1);                        01353
             IF count = lpesc THEN                                     01313
                REPEAT LOOP;                                           01314
             IF (count ← count-40B) <= 0 THEN REPEAT LOOP;             01318
             !pbin();                                                  01321
             *altinp* ← *altinp*, (code ← r1);                         01354
             IF code # irep THEN                                       01322
```

```
                    BEGIN                                                01335
                    WHILE (count ← count-1) DO                           01326
                       BEGIN                                             01327
                       !pbin();                                          01328
                       *altinp* ← *altinp*, r1;                          01329
                       END;                                              01331
                    REPEAT LOOP;                                         01336
                    END;                                                 01337
                 altinp.L ← altinp.L - 3;                                01338
                 !pbin();                                                01332
                 lpxmax ← r1 - 40B;                                      01333
                 !pbin();                                                01339
                 lpymax ← r1 - 40B;                                      01340
                 !pbin();                                                01349
                 lptype ← r1 - 40B;                                      01344
                 !pbin();                                                01350
                 lpdlpad ← r1 - 40B;                                     01345
                 !pbin();                                                01351
                 lpbaudfactor ← r1 - 40B;                                01346
                 EXIT LOOP 2;                                            01324
                 END                                                     01325
              ELSE REPEAT LOOP 2;                                        01308
           END;                                                          01274
        IF i >= 10 THEN %interrogate has not been acknowledged yet%
                                                                         01275
           BEGIN                                                         01276
           typeas($"No Line-Processor data from RLPMD JSYS in
           GETDEV");                                                     01277
           LOOP !haltf;                                                  01278
           END;                                                          01279
        litcolumn[0] ← 1;                                                01285
        !rfmod(100B);                                                    01355
        r2 ←                                                             01356
           r2 .A 7400007777773 .V (lpymax * 2B8) .V (lpxmax * 1B6);
                                                                         01358
        !sfmod( 100B, r2);                                               01357
        r2 ← 4B10 + ((lpdlpad / lpbaudfactor) * 2B8)  % LF padding
        %                                                                01359
           + 4B6  % 2 input buffers %                                    01360
           + ((2 + (8/lpbaudfactor)) * 1B3)  % # ouptut buffers %
                                                                         01361
           + 6B11;  % sending and reading Big Chars %                    01362
        !sfmd2( 100B, r2 );                                              01363
        IF altinp.L THEN &rawchr ← $lpaltgetchar;                        01364
        IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN              01286
           err($"GTJFN failed, GETDEV");                                 01287
        dspjfn ← r1;                                                     01288
        IF NOT SKIP !openf(dspjfn, 1000003B5) THEN                       01289
           err($"OPENF failed, GETDEV");                                 01290
        CASE lptype .A 4M OF                                             01291
           = hazeltine: putbackchar ← TRUE;                              01292
           ENDCASE putbackchar ← FALSE;                                  01293
        !dspjfn ← 0;  %no linked line-processors%                        01294
        END;                                                             01295
  IF devtype = imlac0 OR devtype = imlac1 THEN                            0455
     BEGIN                                                               0456
```

```
      IF NOT SKIP !gtjfn(1B6, ($"TTY:" + 1) .V 18M6) THEN                    0457
         err($"GTJFN failed, GETDEV");                                       0458
      dspjfn ← r1;                                                           0459
      IF NOT SKIP !openf(dspjfn, 1000003B5) THEN                             0460
         err($"OPENF failed, GETDEV");                                       0461
      ldspjfn ← 0;  %no linked line-processors%                              0462
      END;                                                                   0463
   CASE device OF                                                            0464
      = devsri: %user wants DNLS%                                            0465
         CASE devtype OF                                                     0466
            = devsri, = imlac0, = imlac1, = dev1proc: NULL;                  0467
            ENDCASE %not good%                                               0468
               BEGIN                                                         0469
               typeas($"not a display terminal");                            0470
               crlf();                                                       0471
               !JSYS haltf;                                                  0472
               END;                                                          0473
      = devtiex: %user wants TNLS%                                           0474
         CASE devtype OF                                                     0475
            = devsri, = imlac0, = imlac1, = dev1proc:                        0476
               devtype ← devtiex;                                            0477
            ENDCASE;                                                         0478
      = offline:  %user wants DEX%                                           0479
         devtype ← offline;                                                  0480
      ENDCASE; %use devtype%                                                 0481
   CASE devtype OF                                                           0482
      =dev33, =dev35, =dev37, =devtiex, =devsri, =imlac0, =imlac1,
      =nettty, =offline, = dev1proc:                                         0483
         %defined device type%                                               0484
      ENDCASE devtype ← devtiex; %undefined, use devtiex%                    0485
   setdev(devtype);                                                          0486
   RETURN;                                                                   0487
   END.                                                                      0488
                                                                             0489
(setdev) PROCEDURE (device);                                                 0490
   %set up device dependent data for passed device%                          0491
   LOCAL store, i;                                                           0492
   CASE n1device ← device OF                                                 0493
      = devsri:     %our local displays%                                     0494
         BEGIN                                                               0495
         % normal char size: 1; hinc: 12, 14, 18, 20; vinc: 25, 30,
         35, 40; grid: x: 0 to 1023 by y: 256 to 1023;%                      0496
         n1mode ← fulldisplay;                                               0497
         tmarg ← lmarg ← 0;                                                  0498
         rmarg ← 1016*256;                                                   0499
         bmarg ← 1023*256;                                                   0500
         tacsize ← cflcsize ← namcsize ← subcsize ← litcsize ←
         msgcsize ← 1;                                                       0501
         ltcsize ← vscsize ← 0;                                              0502
         tahinc  ← cflhinc ← namhinc ← subhinc ← lithinc ← msghinc ←
         14*256;                                                             0503
         lthinc ← vshinc ← 12*256;                                           0504
         tavinc  ← cflvinc ← namvinc ← subvinc ← litvinc ← msgvinc ←
         ltvinc ← vsvinc ← 30*256;                                           0505
         intdspda(tmarg + 15*tavinc, (bmarg/tavinc)*tavinc, lmarg,
         (rmarg/tahinc)*tahinc);                                             0506
```

```
      %initialize display areas for display terminal%          0507
    END;                                                       0508
= 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
= dev1proc:  %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
```

```
        &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
        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
    littop ← 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 ← tar;                                                   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%                                               062
      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
      littop ← tatop - 2*tavinc;                                       0643
      litbottom ← tabottom;                                            0644
      litleft ← taleft;                                                0645
      litright ← taright;                                              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,
```

```
    netty%                                                              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] ← '\;                                                067
    %backspace word%                                                    0680
        trnsli[27B] ← BW; %↑W%                                          0681
        trnslo[BW] ← '←;                                                0682
    %backspace statement%                                               0683
        trnsli[21B] ← $ascbst; %↑Q%                                     0684
        trnslo[$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
      r2 ← IF nlmode = typewriter THEN ttycoc ELSE dpycoc;             0724
      r3 ← 1B3;                                                        0725
      !JSYS sfcoc;                                                     0726
      END;                                                             0727
   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)%                                                  01371
         IF tenex < 13200 THEN !sbcim(1)                              01367
         ELSE                                                         01368
            BEGIN                                                     01369
            !rfmd2(100B);                                              0730
            !sfmd2( 100B, r2 .V 4B11);                               01366
            END;                                                     01370
      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
            IF NOT SKIP !uasqd(14B10) THEN !JSYS haltf;                0739
      END;                                                             0740
   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, =devsri, =imlac0, =imlac1:         0746
            IF NOT autostrt THEN                                       0747
               BEGIN                                                   0748
               r1 ← 100B;                                              0749
               !JSYS rfmod;                                            0750
               r2 ← r2 .A 777777777717B .V 4B10;                       0751
               !JSYS sfmod;                                            0752
               END;                                                    0753
         ENDCASE;                                                      0754
      END;                                                             0755
   RETURN END.                                                         0756
                                                                       0757
(setdis) PROCEDURE; %set up display areas%
   %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, qa;                                                     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;                                                  01406
      =imlac0, =imlac1: BEGIN                                   0767
         !sout(dspjfn, 4407B8+$rinistr, -11); %send 10 33B's plus
         one other char%                                        0768
         !qisms(1000);                                          0769
         *send* ← begmsg, 1+remfudge, remlcm;                   0770
         !sout(dspjfn, chbmty+$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].rtxl ← [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.datcp, 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 ← $dpyarea) + dal*dacnt;                       0907
         UNTIL &da >= end DO                                       0908
            BEGIN                                                  0909
            IF da.daexis AND NOT da.dasuppress THEN                0910
               <NLS, DSPGEN, alocda>(&da);                         0911
               dassnbit(da.dalsidb, -1, lsbtsize);  %deassign all
               bits%                                               0912
            &da ← &da + dal;                                       0913
            END;                                                   0914
      END;                                                         0915
   ecnoff();                                                       0916
   RETURN;                                                         0917
   END.                                                            0918
                                                                   0919
                                                                   0920
(ttywindow) PROCEDURE (da);  %make a da a tty window%              0921
   LOCAL STRING send[20];                                          0922
   REF da;                                                         0923
   CASE nldevice OF                                                0924
      = devlproc:                                                  0925
         lpttywindow(da.datop, da.dabottom-da.davinc);             0926
      = imlac0, = imlac1: %send out char string%                   0927
         BEGIN                                                     0928
         IF NOT da.dahandle THEN err($"Zero dahandle in ttywindow");
                                                                   0929
         *send* ← begmsg, 4+remfudge, echda, 508, da.dahandle.daidr1
         + remfudge, da.dahandle.daidr2 + remfudge;                0930
         !sout(dspjfn, chbmty+$send, -send.L);                     0931
         END;                                                      0932
      ENDCASE                                                      0933
         %issue uasqd jsys to make the seqda a copy of tty-sim da%
                                                                   0934
            IF NOT SKIP !uasqd(400000001B3 .V da.dahandle) THEN    0935
               err($"UASQD failed, setdis");                       0936
   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.daseq ← 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) + &strl;                            0971
          ls.rtbpe ← chbptr(strl.L) + &strl;                           0972
          IF &str2 THEN &strl ← &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);                                                      0986
    % 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. %                          0987
     LOCAL i, jfn, chain, bp, id[2], wheelf, pcap;                     0988
    LOCAL STRING errorstr[200];                                        0989
    wheelf ← IF (pcap ← enablw()) = -1 THEN 0 ELSE 1;                  0990
    disablw(pcap:=-1);                                                 0991
    jfn ← 0;                                                           0992
    ON SIGNAL ELSE                                                     0993
       BEGIN                                                           0994
       ON SIGNAL ELSE;                                                 0995
```

```
    IF jfn THEN sysclose(jfn);                                          0996
    jfn ← 0;                                                            0997
    END;                                                                0998
  %read id from file or get from user%                                  0999
    IF NOT (jfn ← sgtjfn(1B5, $"<SYSTEM>GROUP.INDEX;1",
    $errorstr)) THEN                                                    01000
      err($"File associating login directories with NLS idents is
      bad.  Please report to ARC personnel.");                          01001
    IF NOT sysopen(jfn, rthawed, bintyp, $errorstr) THEN                01002
      err($"File associating login directories with NLS idents is
      bad.  Please report to ARC personnel.");                          01003
    %get ident from group.index file or user%                           01004
      IF NOT SKIP !sfptr(jfn, user*4) THEN                              01005
        err($"File associating login directories with NLS idents
        is bad.  Please report to ARC personnel.");                     01006
      !bin(jfn); %get user directory number and group number%
                                                                        01007
      IF r2.LH NOT= user THEN                                           01008
        err($"File associating login directories with NLS idents
        is bad.  Please report to ARC personnel.");                     01009
      !bin(jfn); %get first word of ident or link to chain of
      idents%                                                           01010
      IF r2 = 0 THEN                                                    01011
        BEGIN                                                           01012
        IF NOT wheelf THEN err($"no idents associated with this
        login directory");                                              01013
        idfrmuser ($initsr);                                            01014
        END                                                             01015
      ELSE IF r2.LH = 18M THEN %chain of idents%                        01016
        BEGIN                                                           01017
        chain ← r2.RH;                                                  01018
        %get ident from user%                                           01019
          idfrmuser ($initsr);                                          01020
        LOOP                                                            01021
          BEGIN                                                         01022
          IF NOT SKIP !sfptr(jfn, chain) THEN                           01023
            err($"cant set position in file associating login
            directories an idents.  Please report to ARC
            personnel.");                                               01024
          !bin(jfn); %first word of ident%                              01025
          id ← r2;                                                      01026
          !bin(jfn); %second word of ident%                             01027
          id[1] ← r2;                                                   01028
          IF initsr[1] = id AND initsr[2] = id[1] THEN                  01029
            EXIT LOOP; %found a match, everything is ok%                01030
          !bin(jfn); %follow chain to next entry%                       01031
          IF (chain ← r2.RH) = 0 THEN %end of chain%                    01032
            BEGIN                                                       01033
            IF NOT wheelf THEN err($"ident not in list
            associated with this login directory");                     01034
            typeas($"using ident not associated with this
            login directory");                                          01035
            EXIT LOOP;                                                  01036
            END;                                                        01037
          END;                                                          01038
        END                                                             01039
```

```
      ELSE %get the ident%                                              01040
         BEGIN                                                          01041
         initsr[1] ← r2;                                                01042
         !bin(jfn);                                                     01043
         initsr[2] ← r2;                                                01044
         bp ← chbmty + $initsr;                                         01045
         FOR i ← 1 UP UNTIL > 10 DO                                     01046
            IF ↑bp = 0 THEN EXIT LOOP;                                  01047
         initsr.L ← i-1;                                                01048
         END;                                                           01049
   CASE initsr.L OF                                                     01050
      > 4:                                                              01051
         err($"ident too long for use with NLS (max length is 4
         characters)");                                                 01052
      < 1: err($"Illegal ident.  Please report to ARC system
      personnel.");                                                     01053
      ENDCASE;                                                          01054
sysclose(jfn);                                                          01055
jfn ← 0;                                                                01056
%now put initials into cinit, packed into 5-bit%                        01057
   cinit ← setcinit($initsr);                                           01058
RETURN;                                                                 01059
END.
                                                                        01060
(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($"Illegal character in id");                       01070
      CASE count OF                                                     01071
         =1: val.cint1 ← char;                                          01072
         =2: val.cint2 ← char;                                          01073
         =3: val.cint3 ← char;                                          01074
         ENDCASE val.cint4 ← char;                                      01075
      END;                                                              01076
   RETURN (val);                                                        01077
   END.
                                                                        01078
(idfrmuser) PROCEDURE (str);  %collect ident from user in tty mode%
                                                                        01079
   LOCAL char;                                                          01080
   REF str;                                                             01081
   LOOP                                                                 01082
      BEGIN                                                             01083
      typeas($"
      Ident = ");                                                       01084
      *str* ← NULL;                                                     01085
      echon();                                                          01086
      LOOP                                                              01087
         BEGIN                                                          01088
         CASE (char ← rawchr()) OF                                      01089
```

```
            = '?: %help him%                                                01090
               BEGIN                                                        01091
               typeas($"
               If you have already been assigned a unique identifier
               (called an IDENT)");                                         01092
               typeas($" by the ARC Identification system (and you
               remember it), ");                                            01093
               typeas($"then please type it followed by a Carriage
               Return.");                                                   01094
               typeas($"  otherwise, just type the ESC (ALTMODE) key
               on your terminal and contact ARC.
               ");                                                          01095
               REPEAT LOOP 2;                                               01096
               END;                                                         01097
            IN ['a, 'z]: char ← char - 40B;                                 01098
            IN ['A, 'Z], IN ['2, '6]: NULL;                                 01099
            = SP, =BW:  REPEAT LOOP 2;                                      01100
            =CA, =EOL, =CR: EXIT LOOP;                                      01101
            = $ascalt: !haltf;                                              01102
            =BC:                                                            01103
               BEGIN                                                        01104
               IF str.L THEN                                                01105
                  BEGIN                                                     01106
                  typech('/);                                               01107
                  typech(*str*[str.L]);                                     01108
                  str.L ← str.L - 1;                                        01109
                  END;                                                      01110
               REPEAT CASE;                                                 01111
               END;                                                         01112
            ENDCASE                                                         01113
               BEGIN                                                        01114
               typeas($" ? ");                                              01115
               REPEAT CASE;                                                 01116
               END;                                                         01117
         *str* ← *str*, char;                                               01118
         END;                                                               01119
      IF str.L THEN EXIT LOOP;                                              01120
      END;                                                                  01121
   RETURN;                                                                  01122
   END.                                                                     01123
                                                                            01124
(setinterrupts)PROC;                                                        01125
   %set up rubout, command delete, and page write interrupt%                01126
      %IF YOU CHANGE THIS CODE, SEE ALSO (utility, trapcc) AND
      (utility, notrapcc)%                                                  01127
      chntab[1] ← $stopline .V 1B6; % ↑P %                                  01128
      %note: the ↑C dispatch is altered/restored by the journal%
                                                                            01129
      chntab[2] ← $ctrlc .V 1B6; %handle ↑C to do imlac stuff%              01130
      chntab[3] ← $rubout .V 1B6; % treat ↑O like a rubout %                01131
      chntab[4] ← $msgpsi .V 1B6; % for message fork to use %               01132
      chntab[9] ← $stkovr .V 1B6;                                           01133
      chntab[11] ← $iodaterr .V 1B6;                                        01134
      chntab[15] ← $ilspsi .V 1B6;  % illegal instruction %                 01135
      chntab[17] ← $wrtpsi .V 1B6;                                          01136
      chntab[24] ← $gotohelp .V 1B6;                                        01137
```

```
      r1 ← 4B5;                                                          01138
      r2 ← $chntab;                                                      01139
      !HRLI r2, levtab;                                                  01140
      !JSYS sir;                                                         01141
   % special stuff for control-c (↑C)%                                   01142
      r1 ← 4B5;                                                          01143
      !JSYS rpcap;                                                       01144
      r3 ← r3 .V 4B11;                                                   01145
      !JSYS epcap; %Permits process to assign ↑C for pseudo
      interrupt%                                                         01146
   %enable interrupts%                                                   01147
      r1 ← 4B5;                                                          01148
      !JSYS eir;                                                         01149
   %activate ic data error and write pseudo interrupt%                   01150
      r1 ← 4B5;                                                          01151
      r2 ← 105B6; %channel 11, 15, 17%                                   01152
      !JSYS aic;                                                         01153
   %activate rubout, etc.%                                               01154
      r1 ← 200000001B; %↑P%                                              01155
      !JSYS ati;                                                         01156
      r1 ← 170000003B; %↑O%                                              01157
      !JSYS ati;                                                         01158
      r1 ← 210000030B; %↑Q%                                              01159
      !JSYS ati;                                                         01160
      IF nldevice = imlac0 OR nldevice = imlac1 THEN                     01161
         BEGIN                                                           01162
         r1 ← 3000002B; %↑C%                                             01163
         !JSYS ati;                                                      01164
         r2 ← 340000004000B; %activate rubout, ↑P, ↑Q, ↑C, ↑O%           01165
         END                                                             01166
      ELSE  r2 ← 240000004000B; %activate ↑P, ↑Q and ↑O%                 01167
      r1 ← 4B5;                                                          01168
      !JSYS aic;                                                         01169
   RETURN;                                                               01170
   END.
                                                                         01171
(gtintmsg) PROCEDURE;                                                    01172
   typeas($"  Message = ");                                              01173
   LOOP                                                                  01174
      BEGIN                                                              01175
      !pbin;                                                             01176
      IF r1 = CA THEN EXIT;                                              01177
      *intmsg* ← *intmsg*, r1;                                           01178
      END;                                                               01179
   IF intmsg.L THEN                                                      01180
      BEGIN                                                              01181
      *intmsg* ← *intmsg*, 0;                                            01182
      intmsf ← TRUE;                                                     01183
      END;                                                               01184
   RETURN(intmsg.L);                                                     01185
   END.
                                                                         01186
(setabs)    % new bit table version of settabs %                         01187
   PROCEDURE                                                             01188
      (tabtbl);      % address of tab table %                           01189
   LOCAL twrd1, twrd2, twrd3;                                            01190
```

```
    REF tabtbl;                                                          01191
       !gjinf();                % get user info %                        01192
       IF r4 = -1 THEN RETURN;          % NOP if detached %             01193
    % convert NLS internal form to TENEX form %                         01194
       r1 ← tabtbl;                                                      01195
       r2 ← tabtbl[1];                                                   01196
       !SETCA 1,0;              %NLS bits are inverted%                  01197
       !SETCA 2,0;                                                       01198
       twrd1 ← r1;                                                       01201
       twrd2 ← r2;                                                       01202
       r2 ← tabtbl[2];                                                   01204
       !SETCA 2,0;                                                       01206
       twrd3 ← r2;                                                       01208
    % set tabs unless monitor words are the same %                      01209
       !gtabs(18M);                                                      01210
       IF twrd1 = r2 AND twrd2 = r3 AND twrd3 = r4 THEN RETURN           01211
       ELSE !stabs(18M, twrd1, twrd2, twrd3);                            01212
    RETURN;                                                            01213
    END.                                                               01214
                                                                       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
```

FONLY FONLY FONLY FONLY FONLY FONLY FONLY FONLY FONLY
FONLY FONLY FONLY FONLY FONLY FONLY FONLY FONLY FONLY

```
< NLS, FONLY.NLS;4, >, 1-OCT-74 14:35 CHI ;;;;  % 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%                                 01881
         PROCEDURE                                                     01882
            %FORMALS%                                                  01883
               (result,          %result record%                       01884
               parsemode,        %parsing, backup, cleanup%            01885
               window);          %window idnetificatin%                01886
               REF                                                     01887
                  result, window;                                      01888
            LOCAL da; REF da;                                          01889
         %----------------------%                                      01890
         CASE parsemode OF                                             01891
            = parsing:                                                 01892
               BEGIN                                                   01893
               &da ← dsparea(window);                                  01894
               clearda=[&da);                                           01895
               clrall(&da, TRUE);                                      01896
               da.daempty ← TRUE;                                      01897
               da.dacsp ← endfil;                                      01898
               da.daccnt ← 1;                                          01899
               dafrmt(&da, 0);                                         01900
               END;                                                    01901
            ENDCASE;                                                   01902
         RETURN(&result);                                              01903
         END.                                                          01904

   %connect%                                                           01905
      (xconnect) %Execute Connect Command%                             01906
         PROCEDURE                                                     01907
            %FORMALS%                                                  01908
               (result,          %result record%                       01909
               parsemode,        %parsing, backup, cleanup%            01910
               entity,           %entity type%                         01911
               destination,      %destination pointer%                 01912
               parameters);      %password or type of terminal connection%
                                                                       01913
            LOCAL console, tp, dskcn1, dskcn2, dir1, dir2;             01914
            LOCAL STRING locstr[50], erstng[200];                      01915
            REF                                                        01916
               result, entity, destination, parameters, tp;            01917
         %----------------------%                                      01918
         CASE parsemode OF                                             01919
            = parsing:                                                 01920
               CASE entity OF                                          01921
                  = $display:                                          01922
                     BEGIN                                             01923
                     &tp ← &destination+d2sel;                         01924
                     *locstr* ← destination tp;                        01925
                     console ← VALUE($locstr, 8);                      01926
```

```
            %suppress the current display%                          01928
               shutdis();                                           01929
               IF nldevice NOT= devlproc AND NOT SKIP
               !uasqd(14B10) THEN NULL;                             01930
                  %turn tty-simulation off%                         01931
            xconterm(console, IF parameters = $input THEN TRUE
            ELSE FALSE);                                            01927
            LOOP inpt();                                            02346
            END;                                                    01932
         = $tty:                                                    01933
            BEGIN                                                   01934
            &tp ← &destination+d2sel;                               01935
            *locstr* ← destination tp;                              01936
            console ← VALUE($locstr, 8);                            01937
            xconterm(console, IF parameters = $input THEN TRUE
            ELSE FALSE);                                            01938
            IF parameters = $input THEN                             02348
               LOOP inpt();                                         02349
            END;                                                    01939
         = $directory:                                              01940
            BEGIN                                                   01941
            IF parameters = 0 THEN                                  01942
               BEGIN  % no password entered, ensure valid text
               pointers anyway %                                    01943
               FIND SF(*locstr*) ↑parameters;                       01944
               parameters[ d2sel] ← parameters;                     01945
               parameters[ d2sel+1] ← parameters[1];                01946
               END;                                                 01947
            !gjinf();                                               01948
            dir1 ← r2;                                              01949
            !gtdal(0);                                              02360
            dskcn1 ← r2 - r1;                                       02361
            cconfildir(&destination, &destination+d2sel,
            &parameters, &parameters+d2sel);                        01951
            !gjinf();                                               01952
            dir2 ← r2;                                              01953
            !gtdal(0);                                              02362
            dskcn2 ← r2 - r1;                                       02363
            IF dskcn1 > 0 THEN                                      01955
               BEGIN                                                01956
               gdname( dir1, $locstr);                              01957
               *erstng* ←                                           01958
                  EOL, *locstr*, " OVER ALLOCATION BY ",
                  STRING(dskcn1), " PAGES!";                        01959
               END;                                                 01960
            IF (dskcn2 > 0) AND (dir1 # dir2) THEN                  01961
               BEGIN                                                01962
               gdname( dir2, $locstr);                              01963
               *erstng* ←                                           01964
                  *erstng*, EOL, *locstr*, " OVER ALLOCATION BY
                  ", STRING(dskcn2), " PAGES!";                     01965
               END;                                                 01966
            IF erstng.L THEN dismes(2, $erstng);                    01967
            END;                                                    01968
         ENDCASE err(notyet);                                       01969
      ENDCASE;                                                      01970
```

```
      RETURN(&result);                                                    01971
      END.
                                                                          01972
   (xconterm)    %Connect TTY Command%                                    01973
      PROCEDURE(console, inout);                                          01974
      LOCAL i;                                                            01975
      savnldevice ← nldevice;                                             01976
      %arm control-S to remove this connected state%                      01977
         savchntab[1] ← chntab[1] := $brkconnection .V 1B6;               01978
      %refuse Auto-logout%                                                01979
         !atljb(1);                                                       01980
      %advise or link to designated terminal%                             01981
         IF inout THEN %advise%                                           01982
            BEGIN                                                         01983
            FOR i ← 4 DOWN UNTIL <= 0 DO                                  01984
               IF SKIP !adviz(200000B5 .V console) THEN EXIT;             01985
            IF i <= 0 THEN                                                01986
               ERR($"That terminal did not issue Accept Connection
               command");                                                 01987
            END                                                           01988
         ELSE %output linked only%                                        01989
            BEGIN                                                         01990
            IF  NOT SKIP !tlink(14B10 .V 777777B, console .V 400000B)
            THEN                                                          01991
               err($"Unable to connect terminals");                       01992
            END;                                                          01993
      RETURN;                                                             02365
      END.
                                                                          01996
%accept%                                                                  02285
   (xaccept) %Execute Accept Connection Command%                          02286
      PROCEDURE                                                           02287
         %FORMALS%                                                        02288
            (result,          %result record%                           02289
            parsemode,        %parsing, backup, cleanup%                02290
            type,             %input outpout or outputo only%           02354
            console);         %line number of other terminal%           02291
            REF                                                           02292
               result, console#@ type;                                     02293
         LOCAL i, tp, device, save;                                       02294
         LOCAL STRING locstr [10];                                        02295
         REF tp;                                                          02296
      %----------------------%                                           02297
      CASE parsemode OF                                                   02298
         = parsing:                                                       02299
            BEGIN                                                         02300
            device ← nldevice;                                            02301
            savnldevice ← nldevice;                                       02302
            %arm control-S to remove this connected state%                02303
               savchntab[1] ← chntab[1] := $brkconnection .V 1B6;
                                                                          02304
            &tp ← &console+d2sel;                                         02305
            *locstr* ← console tp;                                        02306
            linkcnsl ← VALUE($locstr, 8);                                 02307
            %refuse auto-logout%                                          02308
               !atljb(1);                                                 02309
```

```
            %set receive advise%                                        02319
               IF type = $input THEN                                    02355
                  BEGIN                                                 02356
                  FOR i ← 4 DOWN UNTIL <= 0 DO                          02320
                     IF SKIP !adviz(1000004B5 .V linkcnsl) THEN EXIT
                     LOOP;                                              02321
                  IF i <=  0 THEN                                       02322
                     err ($"That terminal not trying to connect to
                     you");                                             02323
                  END;                                                  02357
               IF nldevice = devlproc THEN                              02310
                  BEGIN                                                 02311
                  save ← enablw();                                      02358
                  *locstr* ← "TTY", console tp, ':, 0;                  02312
                  IF NOT SKIP !gtjfn(1B6, ($locstr+1) .V 18M6) THEN
                                                                        02313
                     BEGIN                                              02342
                     linkcnsl ← 0;                                      02343
                     err($"unable to accept connection -- cant get
                     JFN");                                             02314
                     END;                                               02341
                  ldspjfn ← r1;                                         02315
                  IF NOT SKIP !openf(ldspjfn, 1000003B5) THEN           02316
                     BEGIN                                              02340
                     reljfn(ldspjfn);                                   02345
                     linkcnsl ← ldspjfn ← 0;                            02344
                     err($"unable to accept connection -- cant open tty
                     in binary mode");                                  02317
                     END;                                               02339
                  disablw(save);                                        02359
                  END;                                                  02318
               IF nlmode = fulldisplay THEN                             02351
                  BEGIN                                                 02352
                  %shut down NLS display%                               02324
                     shutdis();                                         02325
                     IF nldevice NOT= devlproc AND NOT SKIP
                     !uasqd(14B10) THEN NULL; %turn tty-simulation off%
                                                                        02326
                  %restore NLS display with proper formating and linked
                  display areas%                                        02327
                     %determine proper formatting%                      02328
                        IF nldevice = devsri AND device NOT= devsri THEN
                                                                        02329
                           setdev(device);                              02330
                     continue ← TRUE;                                   02331
                     <INTNLS, initdis>();                               02332
                     continue ← FALSE;                                  02333
                     alldsp();                                          02334
                  END;                                                  02353
               END;                                                     02335
            ENDCASE;                                                    02336
         RETURN(&result);                                               02337
         END.
                                                                        02338
%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
            IF linkcnsl THEN rstconnection()                           02008
            ELSE err($"You are not connected to anyone");              02009
         ENDCASE;                                                      02010
      RETURN(&result);                                                 02011
      END.                                                             02012

%freeze%                                                               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
               = $frozen:                                           02054
                  BEGIN                                             02055
                  xrelsta (&da, destination);                       02056
                  END;                                              02057
               = $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 %                02092
      % FORMAL ARGUMENTS %                                          02093
         resultptr,          % ptr to result record %               02094
         parsemode);         % interpreter parsing mode %           02095
         REF resultptr;                                             02096
      LOCAL char;                                                   02097
      %----------%                                                  02098
      CASE parsemode OF                                             02099
         = parsing:                                                 02100
            BEGIN                                                   02101
            cspupdate ← FALSE;                                      02102
            echolt();                                               02103
            LOOP                                                    02104
               CASE char ← inpt() OF                                02105
```

```
                    = cachar, = CD:                                    02106
                       BEGIN                                           02107
                       echoff();                                       02108
                       RETURN (&resultptr);                            02109
                       END;                                            02110
                    ENDCASE;                                           02111
                 END;                                                  02112
              ENDCASE;                                                 02113
           END.
                                                                       02114
%.....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
   = $return:                                                       02157
      BEGIN                                                         02158
      % advance through jump stack %                                02159
         readfrring(da.dalink, 0 : &srr);                           02160
         stid ← readsrring(&srr, srrcnt ← srrcnt+1 : cc,
         pvs1, pvs2);                                               02161
            %srrcnt is a global which is set to zero in
            jrinit%                                                 02162
      %display the current statement%                               02163
         % display first 20 chars of stmt in name area %
                                                                    02164
            % get statement length %                               02165
               lodsdb(getsdb(stid):stdb);                           02166
               len ← [stdb].schars + 1; % number of chars
               %                                                    02167
            % construct text ptrs to each end of stmt %
                                                                    02168
               tp1 ← stid;                                          02169
               tp1[1] ← 1;                                          02170
               tp2 ← stid;                                          02171
               tp2[1] ← MIN(20,len); %truncate if over 20
               chars %                                              02172
            % assign something to the string "temp" %
                                                                    02173
               IF len > 1 THEN *temp* ← tp1 tp2 ELSE
               *temp* ← "<NULL>";                                   02174
         dn($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
   = $filereturn:                                                   02182
      BEGIN                                                         02183
      % move through the link stack one positiop,
      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
               = $return: srrcnt ← 0;                               02215
               = $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
```

FREE LUNCH

FRECORDS FRECORDS FRECORDS FRECORDS FRECORDS FRECORDS FRECO
FRECORDS FRECORDS FRECORDS FRECORDS FRECORDS FRECORDS FRECO

```
< NLS, FRECORDS.NLS;2, >, 19-SEP-74 21:41 HGL ;;;;
FILE frecords % L10  <REL-NLS>frecords %% (L10,) (rel-nls,frecords.rel,)
%                                                                    0214
   (initialvalues) RECORD                                            0215
      ileft[36], iright[36], itop[36], ibottom[36], icsize[36],
      ifont[36], ihinc[36], ivinc[36];                               0216
% 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 fbhlpl = 3;                                09
   % 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
         sltpl1[36],    % text ptr to head of selection %           033
         sltpl2[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
      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
      llcmd[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], fzvspc2[36], fzstid[36], fznext[18], fzexis[1];
                                                                       0163
   DECLARE EXTERNAL frz1 = 4;
                                                                       0164
(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
(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
% 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

(MLK)FRONTEND (MLK)FRONTEND (MLK)FRONTEND (MLK)FRONTEND (MLK)FRON

```
< NLS, FRONTEND.NLS;52, >, 16-OCT-74 12:46 DSM ;;;;  % 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
   (freflnt)  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
                     endfrr ← &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

   (filusd)   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
                                                                            0101
```

```
    IF dacnt NOT IN [1,damax] THEN err($"Fatal display error in
    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
         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%                                                  02243
   LOCAL mask; %used to mask out certain viewspecs%                     02244
   REF da;                                                              02245
   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%                                         02246
      BEGIN %determine how to format it%                                02268
      mask ← -1; mask.vslev ← mask.vsrlev ← mask.vslevd ←
      mask.vsbrof ← mask.vsplxf ← 0;                                    02250
         %mask out level fields%                                        02251
      IF (((da.dapvs) .A mask) # ((da.davspec) .A mask))
         %critical viewspecs have just changed%
      OR cdtype = dspallf
      OR da.davspec.vscakf
      OR da.davspec.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 (NOT da.davspec.vsindf AND (da.davspec.vsbrof OR
      da.davspec.vsplxf) AND cdtype NOT= dsprfmt)
      OR (cdtype = dspjpf AND (da.dapstf # da.dacsp.stfile))
      THEN                                                              02252
         RETURN(TRUE, TRUE) %do full reformat%                          02253
      ELSE RETURN(TRUE, FALSE); %do partial reformat%                   02254
      END                                                               02255
   ELSE RETURN(FALSE, FALSE);                                           02256
   END.                                                                 02257
```

```
(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.dafrz1 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.vsindf THEN 'A ELSE 'B;                            0231
    %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.%
   %-------------------%
   PROCEDURE(dpa);
   LOCAL frzelm;
   REF dpa, frzelm;
   IF &frzelm ← dpa.dafrz1 THEN
      BEGIN
      LOOP
         BEGIN
         frzelm.fzexis ← FALSE;
         IF NOT frzelm.fznext THEN EXIT
         ELSE &frzelm ← frzelm.fznext;
         END;
      frzelm.fznext := fzfree := dpa.dafrz1 := 0;
      END;
   RETURN;
   END.

(xfresta)
   %Given the addrss of a display are, an stid, and two
   viewspec words, this routine will create a frozen element
   for the stid passed it, in the display area passed.  If
   the stid is already on the frozen lit for this area, the
   new vspec words will replace the old ones for that element.%
   %-------------------%
   PROCEDURE(dpa, stid, vspec1, vspec2);
   LOCAL frzelm;
   REF dpa, frzelm;
   IF &frzelm ← dpa.dafrz1 THEN
      BEGIN
      LOOP
         BEGIN
         IF frzelm.fzstid = stid THEN
            BEGIN
            frzelm.fzvspec ← vspec1;
            frzelm.fzvspc2 ← vspec2;
            RETURN;
            END;
         IF frzelm.fznext THEN &frzelm ← frzelm.fznext
         ELSE EXIT;
         END;
      IF NOT fzfree THEN err(5);
      &frzelm ← frzelm.fznext ← fzfree := [fzfree].fznext;
      frzelm.fznext ← 0;
      END
   ELSE
      BEGIN
      IF NOT fzfree THEN err(5);
      dpa.dafrz1 ← &frzelm ← fzfree := [fzfree].fznext;
      frzelm.fznext ← 0;
      END;
   frzelm.fzstid ← stid;
   frzelm.fzvspec ← vspec1;
   frzelm.fzvspc2 ← vspec2;
   frzelm.fzexis ← TRUE;
   RETURN;
```

```
    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.dafrz1 ← da.dafrz1 := 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
                  <ININLS, 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.dafrz1 ← da.dafrz1 := 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.dafrz1 ← da.dafrz1 := 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.dafrz1 ← da.dafrz1 := 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 $ge~char 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
      %for 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%
                                                                     0496
      ptr1, %pointer to TP for start of auxilary text%               0497
      ptr2); %pointer to TP for end of auxilary text%                0498
      LOCAL fl;                                                      0499
      REF ptr1, ptr2, fl;                                            0500
       % change dispatch for "rawchr" to auxilary routine %         0501
         auxsav ← &rawchr;  %save current character dispatch%        0502
         &rawchr ← $auxchr; %and subsitute mine(1st time guy)%       0503
      % setup work area and save text pointers %                    0504
         auxwrk ← auxtp1 ← ptr1;                                     0505
         auxwrk[1] ← auxtp1[1] ← ptr1[1];                            0506
         auxtp2 ← ptr2;                                              0507
         auxtp2[1] ← ptr2[1];                                        0508
      % nail down file if not a-string %                            0509
         IF NOT auxwrk.stastr THEN                                   0510
            BEGIN                                                    0511
            &fl ← fintadr(auxwrk.stfile);                            0512
            fl.finoclos ← TRUE;                                      0513
            END;                                                     0514
      % pass over statement names and setup work area %             0515
         IF NOT auxwrk.stastr THEN auxwrk[1] ← fchtxt(getsdb(auxwrk));
                                                                     0516
         fechc1(forward, $auxwrk);                                   0517
      % jam recognition mode to be "demand" %                       0518
         auxmod ← recogmode; %1st save current one %                 0519
         auxmd2 ← recog2mode;                                        0520
         recogmode ← mdemand;                                        0521
         recog2mode ← mdemand;                                       0522
         RETURN;                                                     0523
      END.                                                           0524
   (auxchr) PROCEDURE; %get characters for executable text%          0525
      LOCAL char;                                                    0526
      LOOP                                                           0527
         IF (NOT inptrf) AND (auxwrk # auxtp2 OR POS auxwrk < auxtp2)
         THEN %characters left to get%                               0528
            BEGIN                                                    0529
            %will this make process commands run faster ?%          02554
                r1 ← 0;                                             02555
               !JSYS simch;                                         02556
            % The JSYS makes the fork appear to the scheduler as though
            it were just unblocked for character input%             02558
            ON SIGNAL ELSE auxinterminate();                         0530
            CASE (char ← READC($auxwrk)) OF    %get the next char%   0531
               = ENDCHR:                                             0532
                  IF NOT basestateflag THEN                          0533
```

```
                    BEGIN                                                   0534
                    % continue with normal input if end of statement
                    and in parse or execution%                              0535
                    auxinterminate(); %go terminate auxilary input%
                                                                            0536
                    RETURN(rawchr());                                       0537
                    END                                                     0538
                 ELSE                                                       0539
                    BEGIN                                                   0540
                    auxwrk ← getnxt(auxwrk);                                0541
                    auxwrk[1] ← fchtxt(getsub(auxwrk));                     0542
                    fechcl(forward, $auxwrk);                               0543
                    REPEAT LOOP;                                            0544
                    END                                                     0545
              ENDCASE                                                       0546
                 BEGIN                                                      0547
                 IF echofg THEN typech(char);                               0548
                 RETURN(char);                                              0549
                 END;                                                       0550
           END                                                              0551
        ELSE                                                                0552
           BEGIN                                                            0553
           auxinterminate(); %go terminate auxilary input%                  0554
           IF inptrf THEN                                                   0555
              BEGIN                                                        01817
              dismes(2, $"User Terminated Input");                         01818
              RETURN(CD)                                                   01816
              END                                                          01819
           ELSE RETURN(rawchr());                                          01815
           END;                                                             0556
     END.                                                                   0557
  (auxinterminate) PROCEDURE; %terminate auxilary input%                    0558
     LOCAL fl;                                                              0559
     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
     % 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
           END;                                                             0570
     RETURN;                                                                0571
     END.                                                                   0572
%.....screen splitting Support Routines.....%                               0573
  (vsplit) %split window vertically%
  PROCEDURE (left, cordsl, cords2);                                         0574
     %split the display area (left) vertically, at the column
     determined by cordsl 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
bmoff();                                                             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%                                                 0605
   right.damcol ← right.daright - right.daleft; %use all of
   window width%                                                     0606
   left.damcol ← left.daright - left.daleft; %use all of window
   width%                                                            0607
%update max indentation%                                            02404
   right.damind ← MIN((right.damcol -
   right.daind)/right.daind*right.daind, tamind*right.daind);
                                                                     02405
   left.damind ← MIN((left.damcol -
   left.daind)/left.daind*left.daind, tamind*left.daind);            02406
%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;                        01709
   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(0"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(0"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.daboneighbor ← (&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);                                                      02275
   %returns dano and coordinates for the point on the boundry             02276
   nearest the passed coordinates and the type of boundry (tbound,
                                                                          02277
   bbound, lbound, rbound)%                                               02278
   %--------------------%                                                 02279
   LOCAL                                                                  02280
      da, top, bottom, left, right, dano;                                 02281
   REF da;                                                                02282
   IF dacnt NOT IN /1,damax/ THEN                                         02283
```

```
      err($"Illegal dacnt, boundry");                                    02284
   IF NOT &da ← dsparea(dano ← findda(dacords : dacords.xcord,
   dacords.ycord)) THEN                                                  02285
      err($"findda failed, boundry");                                    02286
   top ← dacords.ycord - da.datop;                                       02288
   bottom ← da.dabottom - dacords.ycord;                                 02290
   left ← dacords.xcord - da.daleft;                                     02292
   right ← da.daright - dacords.xcord;                                   02294
   CASE MIN(left, right, top, bottom) OF                                 02295
      = left:                                                            02297
         BEGIN                                                           02298
         dacords.xcord ← da.daleft;                                      02299
         RETURN(dano, dacords, lbound);                                  02300
            %type of boundry is lbound%                                  02301
         END;                                                            02302
      = right:                                                           02303
         BEGIN                                                           02304
         dacords.xcord ← da.daright;                                     02305
         RETURN(dano, dacords, rbound);                                  02306
            %type of boundry is rbound%                                  02307
         END;                                                            02308
      = top:                                                             02309
         BEGIN                                                           02310
         dacords.ycord ← da.datop;                                       02311
         RETURN(dano, dacords, tbound);                                  02312
            %type of boundry is tbound%                                  02313
         END;                                                            02314
      = bottom:                                                          02315
         BEGIN                                                           02316
         dacords.ycord ← da.dabottom;                                    02317
         RETURN(dano, dacords, bbound);                                  02318
            %type of boundry is bbound%                                  02319
         END;                                                            02320
      ENDCASE;                                                           02321
   END.                                                                  02322
                                                                         02323

(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
```

```
            dal.daleft ← da2.daright ← (ncords.xcord/tahinc)*tahinc;
                                                                    0867
            END                                                     0868
         ELSE  %adjust all da's which share this boundry%           0869
            fixbnd(TRUE, dal.daleft, ncords.xcord, FALSE);          0870
      = rbound: %right boundry of dal, left of da2%                 0871
         IF dal.datop = da2.datop AND                               0872
         dal.dabottom = da2.dabottom THEN                           0873
            BEGIN %just move this boundry%                          0874
            IF ncords.xcord < dal.daleft OR
            (ncords.xcord-dal.daleft)/dal.dahinc < 20 THEN          0875
               ncords.xcord ← dal.daleft + (20*dal.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
            dal.daright ← da2.daleft ← (ncords.xcord/tahinc)*tahinc;
                                                                    0880
            END                                                     0881
         ELSE  %adjust all da's which share this boundry%           0882
            fixbnd(TRUE, dal.daright, ncords.xcord, FALSE);         0883
      ENDCASE;                                                      0884
   fixouf();                                                        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                                                         02365
         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                                                    02407
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 ← $dpyarea;                                           01868
   rt ← 1;                                                   01003
   DO                                                        01004
      IF da.daexis THEN                                      01005
         BEGIN                                               01006
         height ← da.dabottom - da.datop;                    01007
         lines ← height/da.davinc;                           01008
         width ← da.daright-da.daleft;                       01009
         columns ← width/da.dahinc;                          01011
         IF da.darneighbor THEN columns ← columns-3;         01027
         IF da.dabneighbor THEN lines ← lines-1;             01796
         %reapportion display list entries%                  01022
            freelsrt($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
(lcfile) %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 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(0"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(0"Illegal file return ring");            01275
   % zero file number of current top SRRING %                          01276
      &fre ← &frr + frrhlen + (frrelen*frr.frhtop);                     01277
      IF fR.frexis THEN                                                 01278
         BEGIN                                                          01279
         &srr ← frr.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
(frrlength)  %return number of entries in the file return ring.%
PROCEDURE (frr);                                                01337
   LOCAL i, j, 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
      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(0);                                                       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(0);                                                    01370
           END;                                                             01371
      END;                                                                  01372
   RETURN(i);                                                               01373
   END.                                                                     01374
                                                                            01375
(updsfnm) %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;                                              01379
   REF ofn, nfn, da, str;                                                   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);                                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
                  END;                                                      01392
               *str* ← *nfn*;                                               01393
               END;                                                         01394
            END;                                                            01800
         END;                                                               01809
   RETURN;                                                                  01395
   END.                                                                     01396
```

```
(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);                            01455
   LOCAL sre; REF sre;                                                 01456
   REF srr;                                                            01457
   %verify file number%                                                01458
      IF srr.srhfileno AND stid.stfile NOT= srr.srhfileno THEN         01459
         err(@"file numbers do not match in storesrring");             01460
   %get address of top entry%                                          01461
      &sre ← &srr + srrhlen + (srrelen*srr.srhtop);                    01462
   %store new values%                                                  01463
      sre.srpsid ← stid.stpsid;                                        01464
      sre.srcc ← cc;                                                   01465
      sre.srvs1 ← vs1;                                                 01466
      sre.srvs2 ← vs2;                                                 01467
      sre.srexis ← TRUE;                                               01468
   RETURN;                                                             01469
   END.                                                                01470
(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.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%                                   01565
        r1 ← 4B5; r2 ← 24B10; !JSYS dic;                                01566
        r1 ← 15;  !JSYS 140B;  %deassign terminal code%                 01567
        r1 ← 19;  !JSYS 140B;  %deassign terminal code%                 01568
    LOOP                                                                01569
        CASE char ← input() OF                                          01570
            =EC %↑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
```

```
    $workl ← MAX(1, $workl-1);                                    01602
    fechcl(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 %cther 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;                                                 01661
   %reassign the terminal codes%                                    01662
      r1 ← 23000001B; %↑S%                                          01663
      !JSYS ati;                                                    01664
      r1 ← 17000003B; %↑O%                                          01665
      !JSYS ati;                                                    01666
   r1 ← 4B5;                                                        01667
   r2 ← 24B10; %activate ↑S, ↑O%                                    01668
   !JSYS aic;                                                       01669
   RETURN END.
                                                                    01670
(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: RETURN(FALSE);                                   01693
         = target:                                                  01694
            BEGIN                                                   01695
            swork[1] ← save;                                        01696
            fechcl(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
```

IVENT SUPPORT

(MLK)IDENTSUPPORT (MLK)IDENTSUPPORT (MLK)IDENTSUPPORT (MLK)IDENTSUPP

```
< NLS, IDENTSUPPORT.NLS;5, >, 25-OCT-74 10:10 KJM ;;;;( NLS,
CIDENT.NLS;12, ), 23-MAY-74 13:54 HGL ;
FILE identsupport % L10 to <REL-NLS>identsupport %% (L10,)
(rel-nls,identsupport.rel,) %                                          02
%....declarations....%                                                 03
   DECLARE EXTERNAL lname=1, strinname=2, idchr=3, afgptyp=4;          06
   DECLARE EXTERNAL STRING nwidstr = "NEWIDS", nullfield = "NONE"; 02781
%.....load and nail down idents.master file.....%                      02780
   (loaidfil) PROCEDURE;  %load the Master Ident File Read Only%       02773
      LOCAL fileno;                                                    02774
      fileno ← open(0, jflname($"identfile"));                         02775
      (fintadr(fileno)).flnoclos ← TRUE;                               02776
      RETURN(fileno);                                                  02777
      END.                                                             02778
                                                                       02779
%....check idents against ident-file....%                              08
   (cknlsid) %validate ident for nls user%
   PROCEDURE (identsr, infostr, fileno);                               09
      %This procedure is used to validate idents for use in entering
      nls.  Currently it assumes that any ident that isn't a group or
      organization is legal%                                           010
      %----------------%                                              011
      IF NOT infostr THEN                                              012
         IF NOT oldid(identsr, fileno) THEN RETURN(FALSE)              013
         ELSE NULL                                                     014
      ELSE                                                             015
         IF NOT ckident(identsr, infostr, fileno) THEN RETURN(FALSE);
                                                                       016
      IF orgrptst(identsr, 0) THEN RETURN(FALSE);                      017
      RETURN(TRUE);                                                    018
      END.
                                                                       019
   (ckident) %validate an ident and fetch entry if desired%
   PROCEDURE (identsr, retinfo, fileno);                           02660
      LOCAL idstid, retval, stid, count;                           02661
      LOCAL TEXT POINTER tp1, tp2;                                 02662
      REF identsr, retinfo, lgngrps;                               02663
      %This routine checks the validity of the ident in identsr, and
      returns the information concerning that person from he systems
      ident file in the retinfo string.  If fileno is non-zero, it is
      assumed to be the file number of the ident file...otherwise, the
      ident file is opened%                                        02664
      %If a valid ident, also returns the stid of the statement which
      corresponds to the ident.%                                   02665
                                                                   02666
      IF identsr.L = empty THEN RETURN(FALSE);                     02667
      idstid ← orgstid;                                            02668
      IF NOT fileno THEN                                           02669
         BEGIN                                                     02670
         ON SIGNAL ELSE close(idstid.stfile := 0);                 02671
         idstid.stfile ← open(0, jflname($"identfile"));           02672
         END                                                       02673
      ELSE                                                         02674
         idstid.stfile ← fileno;                                   02675
      astruc(&identsr);                                            02676
      IF (retval ← ((stid ← namelook(idstid, &identsr)) # endfil)) THEN
```

```
                                                                    02677
      BEGIN                                                         02678
      IF NOT (FIND SF(stid) [')] $SP "LAST NAME") THEN              02679
         BEGIN                                                      02680
         IF &retinfo > 0 THEN                                       02681
            BEGIN                                                   02682
            *retinfo* ← SF(stid) SE(stid);                          02683
            IF NOT &lgngrps AND *identsr* = *initsr* THEN           02684
               BEGIN                                                02685
               getigrps (&retinfo, 0, $tp1, $tp2);                  02686
               IF (count ← tp2 [1] - tp1 [1]) THEN                  02687
                  BEGIN                                             02688
                  &lgngrps ← getstring (count, $dspblk);            02689
                  *lgngrps* ← tp1 tp2;                              02690
                  END                                               02691
               ELSE &lgngrps ← $" ";                                02692
               END;                                                 02693
            END;                                                    02694
         END                                                        02695
      ELSE retval ← FALSE;                                          02696
      END;                                                          02697
   IF NOT fileno THEN close(idstid.stfile := 0);                    02698
   RETURN (retval, stid);                                           02699
   END.                                                             02700

(ckidlist) %check if list of idents are already being used%
PROCEDURE (idlist, badidlist, fileno);                              047
   %this routine assumes an identlist in IDLIST and returns with
   IDLIST containing a list of valid (in use) Idents and appends to
   BADIDLIST a list of those idents which are not now in use.%      048
   LOCAL startstid, idstid, stid;                                   049
   LOCAL TEXT POINTER z1, z2, z3, z4, c1, c2;                       050
   LOCAL STRING                                                     051
      identsr[50], work[50], comment[150], special[10],
      origlist[500];                                                02410
   REF idlist, badidlist;                                           052
   %-----------------%                                             02409
   IF idlist.L = empty THEN RETURN;                                 054
   IF badidlist.L THEN *badidlist* ← *badidlist*, SP;               053
   %If fileno is non-zero, it is assumed to be the file number of
   the ident file...otherwise, the ident file is opened%            055
      idstid ← orgstid;                                             056
      IF NOT fileno THEN                                            057
         BEGIN                                                      058
         ON SIGNAL ELSE IF idstid.stfile THEN close(idstid.stfile :=
         0);                                                        059
         idstid.stfile ← open(0, jfname($"identfile"));             060
         END                                                        061
      ELSE                                                          062
         idstid.stfile ← fileno;                                    063
   *origlist* ← *idlist*;                                           064
   *idlist* ← NULL;                                                 065
   FIND SF(*origlist*) TC2;                                         066
   IF (startstid ← namelook(idstid, $"'usedids'")) NOT = endfil AND
   (startstid := getsub(startstid)) NOT= startstid THEN             067
      LOOP  %process each ident in origidlist%                      068
```

```
        BEGIN                                                          069
        FIND c2 $(SP/',) ↑z3 $1('↑/'&) ↑z4 ↑z1 $(LD/'-) ↑z2 $SP ('(
        ↑c1 ←c1 [')] ↑c2 / ↑c1 ↑c2);                                   070
        *identsr* ← +z1 z2;                                            071
        IF NOT identsr.L THEN EXIT; %no more idents%                   072
        *comment* ← c1 c2;                                            02404
        *special* ← z3 z4;                                            02407
        *work* ←',, *identsr*, ',;                                     073
        stid ← startstid;                                              074
        LOOP  %scan the list of assigned idents to find this one%
                                                                       075
           BEGIN                                                       076
           FIND SF(stid) ↑z1;                                          077
           WHILE ( FIND z1 [*identsr*] ↑z1 ) DO                        078
              IF FIND < [',] > *work* THEN                             079
                 BEGIN                                                 080
                 *idlist* ← *idlist*, *special*, *identsr*,
                 *comment*, SP;                                        081
                 EXIT LOOP 2;                                          082
                 END;                                                  083
           IF getfil(stid) THEN                                        084
              BEGIN                                                    085
              *badidlist* ← *badidlist*, *special*, *identsr*,
              *comment*, SP;                                           086
              EXIT LOOP;                                               087
              END;                                                     088
           stid ← getsuc(stid);                                        089
           END;                                                        090
        END;                                                           091
   IF NOT fileno THEN close(idstid.stfile := 0);                       092
   IF idlist.L THEN BUMPDOWN idlist.L;                                02405
   IF badidlist.L THEN BUMPDOWN badidlist.L;                          02406
   RETURN;                                                             093
   END.                                                                094
                                                                      02408

(oldid) %check if an ident is already being used%
PROCEDURE (identsr, fileno);                                           095
   %this routine returns true if the ident passed is aready in use,
   false otherwise -- much faster than ckident%                        096
   LOCAL idstid, stid;                                                 097
   LOCAL TEXT POINTER z1;                                              098
   LOCAL STRING work[200];                                             099
   REF identsr;                                                       0100
   %If fileno is non-zero, it is assumed to be the file number of
   the ident file...otherwise, the ident file is opened%              0101
      IF identsr.L = empty THEN RETURN(FALSE);                        0103
      idstid ← orgstid;                                               0102
      IF NOT fileno THEN                                              0104
         BEGIN                                                        0105
         ON SIGNAL ELSE close(idstid.stfile := 0);                    0106
         idstid?stfile ← open(0, jfname($"identfile"));               0107
         END                                                          0108
      ELSE                                                            0109
         idstid.stfile ← fileno;                                      0110
   astruc(&identsr);                                                  0111
   *work* ←',, *identsr*, ',;                                         0112
```

```
    IF (stid ← namelook(idstid, $"'usedids'")) NOT = endfil AND (stid
    := getsub(stid)) NOT= stid THEN                                    0113
       LOOP                                                            0114
          BEGIN                                                        0115
          FIND SF(stid) ↑z1;                                           0116
          WHILE ( FIND z1 /*identsr*/ ↑z1 ) DO                         0117
             IF FIND < /',/ > *work* THEN                              0118
                BEGIN                                                  0119
                IF NOT fileno THEN close(idstid.stfile := 0);          0120
                RETURN(TRUE);                                          0121
                END;                                                   0122
          IF getftl(stid) THEN EXIT LOOP;                              0123
          stid ← getsuc(stid);                                         0124
          END;                                                         0125
    IF NOT fileno THEN close(idstid.stfile := 0);                      0126
    RETURN(FALSE);                                                     0127
    END.                                                               0128
%....Retrieve logical fields from ident-file entry....%                0245
    (ldelivery) %get LOGICAL DELIVERY TYPE(S) for an ident entry%
    PROCEDURE (stid);                                                  0246
       %stid points to identification record.                          0247
          Returns record of delivery types indicated in/by record (see
          record definition 'deliverymode' in utilty for possible
          types)%                                                      0248
       LOCAL retval, oldelf, arcorgf, astrng;                          0249
       LOCAL TEXT POINTER z1, z2, z3, z4;                              0251
       LOCAL STRING tempsr/50/;                                        0250
       REF astrng;                                                    02701
       &astrng ← 0;                                                   02704
       ON SIGNAL ELSE                                                 02722
          IF &astrng THEN freestring(&astrng:=0, $dspblk);            02723
       &astrng ← getstring(2000, $dspblk);                            02705
       IF NOT stid.stastr THEN                                         0253
          BEGIN                                                        0254
          *astrng* ← SF(stid) SE(stid);                                0255
          stid ← asrref(&astrng);                                      0256
          END;                                                         0257
       getidelivery(stid.stadr, 0, $z1, $z2);                          0258
       retval ← 0;                                                     0259
       retval.delhc ← FIND BETWEEN z1 z2(/"Hardcopy"//"Hard Copy"/);
                                                                       0260
       retval.delnet ← FIND BETWEEN z1 z2(/"Network"//"Net Work"/);    0261
       oldelf ← FIND BETWEEN z1 z2(/"Online"//"On-Line"/);             0262
       IF retval = 0 OR oldelf THEN                                    0263
          BEGIN                                                        0264
          getiorg(stid.stadr, $tempsr, 0, 0);                          0265
          arcorgf ← IF *tempsr* = "SRI-ARC" THEN TRUE ELSE FALSE;      0266
          END;                                                         0267
       IF retval = 0 AND NOT oldelf THEN                               0268
          BEGIN                                                        0269
          %figure out a default delivery%                              0270
          %NLS users and ARC members get On-Line%                      0271
             IF arcorgf THEN oldelf ← TRUE                             0272
             ELSE                                                      0273
                BEGIN                                                  0274
                %everyone gets hard copy now%                          0275
```

```
            retval.delhc ← TRUE;                                    0276
         geticapability(stid.stadr, 0, $z1, $z2);                   0277
         oldelf ← FIND BETWEEN z1 z2( (\NLS"));                     0278
         END;                                                       0279
      END;                                                          0280
   IF oldelf THEN                                                   0281
      BEGIN                                                         0282
      oldelf ← IF arcorgf THEN 2 ELSE 1; %set default nlhost%       0283
      getinlhost(stid.stadr, 0, $z3, $z4);                          0284
      IF z3[1] # z4[1] THEN                                         0285
         BEGIN                                                      0286
         oldelf ← 0;                                                0287
         IF FIND BETWEEN z3 z4 ([*arcstr*]) THEN oldelf ← oldelf .V
         2;                                                         0288
         IF FIND BETWEEN z3 z4 ([*utilstr*]) THEN oldelf ← oldelf .V
         1;                                                         0289
         END;                                                       0290
      oldelf ← oldelf .A (CASE lhostn OF                            0291
         =archost: 2B;                                              0292
         =utilhost: 1B;                                             0293
         ENDCASE 0);                                                0294
      IF oldelf THEN retval.delcl ← TRUE;                           0295
      END;                                                          0296
   IF &astrng THEN freestring(&astrng:=0, $dspblk);                02724
   RETURN(retval);                                                  0297
   END.
                                                                    0298
(luser) %get LOGICAL USER-NAME (for NLS delivery) for an ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                           0299
   LOCAL TEXT POINTER lptr1, lptr2;                                 0300
   LOCAL retval;                                                    0301
   REF entrystr;                                                    0302
   retval ← FALSE;                                                  0303
   IF NOT                                                           0304
      (FIND SF(*entrystr*) ["User:"]                                0305
         $NP  ↑lptr1 [';] < CH $NP > ↑lptr2) THEN                   0306
            getilname(&entrystr, fldstr, ptr1, ptr2)                0307
   ELSE                                                             0308
      BEGIN                                                         0309
      stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                  0310
      retval ← TRUE;                                                0311
      END;                                                          0312
   RETURN(retval);                                                  0313
   END.
                                                                    0314
(lgetsubcoll) %get LOGICAL SUBCOLLECTIONS for an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                           0315
   LOCAL TEXT POINTER lptr1, lptr2;                                 0316
   LOCAL retval;                                                    0317
   LOCAL STRING tempsr[30];                                         0318
   REF entrystr;                                                    0319
   IF NOT (retval ← getisubcol(&entrystr, fldstr, ptr1, ptr2)) THEN
                                                                    0320
      IF orgrptst(&entrystr, 0) THEN getiid(&entrystr, fldstr, ptr1,
      ptr2)                                                         0321
```

```
      ELSE                                                              0322
         BEGIN                                                          0323
         getiorg(&entrystr, $tempsr, 0, 0);                             0324
         IF *tempsr* # "SRI-ARC" THEN *tempsr* ← "NIC";                 0325
         FIND SF(*tempsr*) ↑1ptr1 SE(*tempsr*) > ↑1ptr2;                0326
         stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                   0327
         END;                                                           0328
   RETURN(retval);                                                      0329
   END.
                                                                        0330
(laddress) %get LOGICAL U.S. MAIL ADDRESS for an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2, fileno);                       0331
   % Fileno is the file number of an open file in which the address
   ident is checked;  it is passed to ckident.  If zero, the
   identfile is used and opened.  Entrystr is the address of a
   string containing the text of the ident entry whose mailing
   address is sought; fldstr is the address of a strig into which
   the mailing address is placed.  ptr1 and ptr2 are either the
   address of two text pointers which will delimit the  mailing
   address in the field string passed; they may be zero.  The
   procedure returns false if the address is not found. %       0332
   %Get the mailing address, even if you nave to go to another ident
   for it -- New Version by WLB 5 July 1972; modified by HGL
   18-DEC-73%                                                           0333
   LOCAL retval;                                                        0334
   LOCAL TEXT POINTER tpf, tpe;                                         0335
   LOCAL STRING savedid[30];                                            0336
   REF fldstr, entrystr, ptr1, ptr2;                                    0337
   IF NOT &fldstr THEN RETURN (FALSE);                                  0338
   getiadd(&entrystr, &fldstr, &ptr1, &ptr2);                           0339
   IF NOT FIND SF(*fldstr*) [NP] THEN                                   0340
      BEGIN %See if it is an ident%                                     0341
      % save away this ident:  the entrystr will be smashed and then
      restored using it.  This is done to avoid using space.  It is
      slow and could be avoided by using another string. %          0342
      getiid (&entrystr, $savedid, 0, 0);                               0343
      IF ckident(&fldstr, &entrystr, fileno) THEN                       0344
         BEGIN % entrystr now has the info for the ident (a group or
         person) which has the address. %                           0345
         retval ← laddress(&entrystr, &fldstr, &ptr1, &ptr2,
         fileno);                                                   0346
         getinam (&entrystr, 0, $tpf, $tpe);                        0347
         IF orgtst (&entrystr, 0) THEN % Must have organization name
         at beginning %                                             0348
            *fldstr* ← tpf tpe, EOL, *fldstr*                       0349
         ELSE *fldstr* ← "c/o ", tpf tpe, EOL, *fldstr*;            0350
         % Now restore original string. %                           0351
         IF ckident ( $savedid, &entrystr, fileno ) AND retval THEN
                                                                    0352
            BEGIN                                                   0353
            IF &ptr1 THEN FIND SF(*fldstr*) ↑ptr1;                  0354
            IF &ptr2 THEN FIND SE(*fldstr*) ↑ptr2;                  0355
            RETURN(TRUE);                                           0356
            END                                                     0357
         ELSE RETURN( FALSE);                                       0358
         END;                                                       0359
```

```
    IF NOT ckident ( $savedid, &entrystr, fileno ) THEN RETURN
    (FALSE); % Return FALSE if we could not restore the original
    information %                                                    0360
    END;                                                            0361
  IF &ptr1 THEN FIND SF(*fldstr*) ↑ptr1;                            0362
  IF &ptr2 THEN FIND SE(*fldstr*) ↑ptr2;                            0363
  RETURN (TRUE);                                                    0364
  END.
                                                                    0365
(lmemlist) %get LOGICAL MEMBERSHIP for a group/organization ident%
PROCEDURE (entrystr, outlist, explist, noexplist, errorlist,
idfile);                                                            0366
  % Gets the logical membership list of a group or organization --
  i.e., expands membership list (recursively) until only individual
  idents are present in list.  This routine is supposed to
  correctly handle expand/noexpand issues and also will not loop in
  the event of circular references -- e.g., GROUPA is a member of
  GROUPB is a member of GROUPC is a member of GROUPA -- isn't that
  a cute feature!                                                   0367
    ENTRYSTR is the name of a string containing the text of an
    identfile entry (as copied from the identfile) -- the name,
    coordinator, and memlist of the group/organization being
    referenced will be extracted from this string.                  0368
      NOTE: this must be a legitimate L10 STRING, as it will be
      used for working space by lmemlist -- the original contents
      will be restored before lmemlist returns, however.            0369
    OUTLIST is the name of a string in which the logically
    expanded memlist will be returned -- it must be large enough
    to hold the requested results!                                  0370
      In the output list each individual's ident is followed by a
      parenthetical expression containing a list of his
      "capacities" seperated by spaces.                             0371
      A person's capacity is the ident of a group or organization
      in whose memlist his ident appears.                           0372
        E.g., if GROUPC is a member of GROUPB, GROUPB is a
        member of GROUPA, and JOEBLOW is a member of GROUPC,
        then JOEBLOW's capacity is GROUPC.                          0373
      If an individual has more than one capacity in a group,
      then all will be listed in the parenthetical expression
      following his ident in the output list -- i.e., his ident
      itself will appear only once.                                 0374
        E.g., if GROUPC is a member of GROUPB, GROUPB is a
        member of GROUPA, and JOEBLOW is a member of GROUPA and
        GROUPC, then JOEBLOW's part of OUTLIST would be:            0375
          JOEBLOW(GROUPA GROUPC)                                    0376
      The first ident listed in OUTLIST will be that of the
      coordinator of the group/organization whose memlist is
      being expanded.                                               0377
    EXPLIST is the name of a string in which will be returned a
    list of idents (each followed by a space) of all
    groups/organizations which have been expanded in the course of
    producing OUTLIST  -- i.e., if an ident appears as a capacity
    on OUTLIST it will appear as an entry on EXPLIST.               0378
    NOEXPLIST is the name of a string in which will be returned a
    list of idents (each followed by a space) of all
    groups/organizations which have been encountered but not
```

```
    expanded in the course of producing OUTLIST -- i.e., idents
    which were found preceeded by '& and idents for no-expand
    groups/organizations which were not found preceeded by '↑.  0379
       NOTE that no group/organization which is found on NOEXPLIST
       will be found as a capacity on OUTLIST -- if a
       group/organization appears in both an expand and a
       no-expand context, the expand context takes precedent.  0380
    ERRORLIST is the name of a string in which will be returned a
    list of idents (each followed by a space) which were
    encountered in the course of producing OUTLIST and which could
    not be found in the identfile.                              0381
    IDFILE is the file number of the identfile, if open when
    lmemlist was called, or zero.                               0382
    %                                                           0383
LOCAL lmlflag, expchr;                                          0384
LOCAL TEXT POINTER p1, p2, p3, p4, inp, expp, noexpp;           0385
LOCAL STRING inlist[500], inid[20], capacity[20], idstr[20];    0386
REF entrystr, outlist, explist, noexplist, errorlist;           0387
IF NOT orgrptst (&entrystr,0) THEN RETURN(FALSE);               0388
% Initialize Group Scan %                                       0389
    getiid (&entrystr, 0, $p1, $p2); *inid* ← +p1 p2;           0390
    *explist* ← *inid*,.SP;                                     0391
    FIND SF(*explist*) ↑expp;                                   0392
    *outlist* ← NULL;                                           0393
    *noexplist* ← NULL;                                         0394
    *errorlist* ← NULL;                                         0395
    lmlflag ← TRUE;                                             0396
LOOP BEGIN                                                      0397
    IF NOT FIND expp > ↑p1 (SP) ↑expp ↑p2 ←p2 THEN EXIT LOOP;   0398
    *capacity* ← p1 p2;                                         0399
    IF lmlflag                                                  0400
       THEN lmlflag ← FALSE                                     0401
       ELSE IF NOT ckident ($capacity, &entrystr, idfile)       0402
          THEN BEGIN                                            0403
             *errorlist* ← *errorlist*, *capacity*, SP;         0404
             REPEAT LOOP;                                       0405
             END;                                               0406
    getimem (&entrystr, 0, $p1, $p2); *inlist* ← +p1 p2;        0407
    geticord (&entrystr, 0, $p1, $p2); *idstr* ← +p1 p2;        0408
    WHILE ( FIND SE(*inlist*) '; ) DO BUMPDOWN inlist.L;        0409
    IF idstr.L # 0 AND FIND SF(*inlist*) ( *idstr* ↑p2 -(LD/'-) <
    p2 1$(LD/'-) $SP ↑p1 > $SP *idstr* )                        0410
       THEN *inlist* ← *idstr*, SP, SF(*inlist*) p1, p2
       SE(*inlist*)                                             0411
       ELSE *inlist* ← *idstr*, SP, *inlist*;                   0412
       % INLIST now has mem list with coordinator's id at front %
                                                                0413
    FIND SF(*inlist*) ↑inp;                                     0414
    LOOP BEGIN                                                  0415
       FIND inp > $( $NP ↑inp '( (') );                         0416
       expchr ← IF FIND inp '↑ $NP ↑inp THEN '↑                 0417
          ELSE IF FIND inp '& $NP ↑inp THEN '&                  0418
          ELSE 0;                                               0419
       IF NOT FIND inp ↑p1 L $(LD/'-) ↑p2 ↑inp THEN EXIT LOOP;  0420
       *idstr* ← p1 p2;                                         0421
       IF NOT ckident ($idstr, &entrystr, idfile)               0422
```

```
            THEN BEGIN                                                  0423
               *errorlist* ← *errorlist*, *idstr*, '(, *capacity*,
               '), SP;                                                  0424
               REPEAT LOOP;                                             0425
               END;                                                     0426
         IF crerptst (&entrystr, 0)                                     0427
            THEN BEGIN                                                  0428
               IF FIND SF(*explist*) ( *idstr* SP / [ SP *idstr* SP
               ] )                                                      0429
                  THEN NULL                                             0430
                  ELSE CASE expchr OF                                   0431
                     = '↑ : BEGIN                                       0432
                        *explist* ← *explist*, *idstr*, SP;             0433
                        END;                                            0434
                     = '& : IF NOT FIND SF(*noexplist*) ( *idstr* SP
                     / [ SP *idstr* SP ] )                              0435
                        THEN *ncexplist* ← *noexplist*, *idstr*, SP;
                                                                        0436
                     ENDCASE IF expdtst (&entrystr, 0)                  0437
                        THEN REPEAT CASE ('↑)                           0438
                        ELSE REPEAT CASE ('&);                          0439
               END                                                      0440
            ELSE IF FIND SF(*outlist*) ( *idstr* '( / [ SP *idstr*
            '( ] ) [')] ↑p1 ←p1                                         0441
               THEN *outlist* ←                                         0442
                  SF(*outlist*) p1, SP, *capacity*, p1 SE(*outlist*)
                                                                        0443
               ELSE *outlist* ←                                         0444
                  *outlist*, *idstr*, '(, *capacity*, '), SP;           0445
         END; %LOOP%                                                    0446
      END; %LOOP%                                                       0447
   FIND SF(*noexplist*) ↑noexpp;                                        0448
   lmlflag ← FALSE;                                                     0449
   WHILE ( FIND noexpp ↑p1 [SP] ↑p2 ←p2 ↑noexpp) DO BEGIN               0450
      *capacity* ← p1 p2;                                               0451
      IF FIND SF(*explist*) ( *capacity* SP / [ SP *capacity* SP ] )
                                                                        0452
         THEN BEGIN                                                     0453
            FIND noexpp < ↑p1 SP 1$(LD/'-) ↑noexpp;                     0454
            *noexplist* ← SF(*ncexplist*) noexpp, p1 SE(*noexpp*);
                                                                        0455
            END;                                                        0456
      END;                                                              0457
   ckident ($inid, &entrystr, idfile); % Restore entrystr %             0458
   RETURN (TRUE);                                                       0459
   END.                                                                 0460
%....retrieve fields from entry in ident-file....%                      0461
   (getiid) %get IDENT field from an ident entry%
   PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0462
      LOCAL TEXT POINTER lptr1, lptr2;                                  0463
      REF entrystr;                                                     0464
      FIND SF(*entrystr*) $(SP/TAB) '( $NP ↑lptr1 [')] <CH $NP> ↑lptr2;
                                                                        0465
      stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                      0466
      RETURN (TRUE);                                                    0467
      END.
```

```
                                                                        0468
(getiadd) %get MAIL ADDRESS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0469
   LOCAL TEXT POINTER lptr1, lptr2;                                     0470
   REF entrystr;                                                        0471
   IF orgrptst(&entrystr, 0) THEN                                       0472
      FIND SF(*entrystr*) 4[EOL] $SP ↑lptr1                             0473
   ELSE                                                                 0474
      FIND SF(*entrystr*) 2[EOL] ↑lptr1;                                0475
   IF NOT (FIND lptr1 [EOL EOL] < $NP ↑lptr2 >) THEN                    0476
      FIND lptr1 > ↑lptr2;                                              0477
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                         0478
   RETURN (TRUE);                                                       0479
   END.
                                                                        0480
(getinam) %get NAME field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0481
   LOCAL TEXT POINTER lptr1, lptr2;                                     0482
   REF entrystr;                                                        0483
   IF orgrptst(&entrystr, 0) THEN                                       0484
      FIND SF(*entrystr*) 2[EOL] $SP ↑lptr1                             0485
   ELSE                                                                 0486
      FIND SF(*entrystr*) [EOL] $SP ↑lptr1;                             0487
   FIND lptr1 [EOL] < $NP ↑lptr2 >;                                     0488
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                         0489
   RETURN (TRUE);                                                       0490
   END.
                                                                        0491
(getifnf) %get NAME field (FIRST NAME FIRST) field from an
individual ident entry%
PROCEDURE (entrystr, fldstr); %first name first%                        0492
   LOCAL grpflg;                                                        0493
   LOCAL TEXT POINTER lptr1, lptr2, lptr3, lptr4;                       0494
   REF entrystr, fldstr;                                                0495
   IF (grpflg ← orgrptst(&entrystr, 0)) THEN                            0496
      FIND SF(*entrystr*) 2[EOL]                                        0497
   ELSE                                                                 0498
      FIND SF(*entrystr*) [EOL];                                        0499
   FIND $SP ↑lptr1 ↑lptr2 ↑lptr3 [EOL] < $NP ↑lptr4 >;                  0500
   IF NOT grpflg THEN FIND BETWEEN lptr1 lptr4 ([',] $NP ↑lptr3 <
   $NP CH $NP ↑lptr2);                                                  0501
   *fldstr* ← lptr3 lptr4, SP, lptr1 lptr2;                             0502
   RETURN (TRUE);                                                       0503
   END.
                                                                        0504
(getilname) %get LAST NAME field from an individual ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                               0505
   %Get last name%                                                      0506
   LOCAL TEXT POINTER lptr1, lptr2;                                     0507
   REF entrystr;                                                        0508
   getinam(&entrystr, 0, $lptr1, $lptr2);                               0509
   FIND lptr1 > [',] ↑lptr2←lptr2;                                      0510
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                         0511
   RETURN (TRUE);                                                       0512
   END.
                                                                        0513
```

```
(geticrg) %get ORGANIZATION field from an individual ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                  03073
   LOCAL TEXT POINTER lptr1, lptr2;                                        03074
   REF entrystr;                                                           03075
   IF orgrptst(&entrystr, 0)                                               03076
   OR NOT
      (FIND SF(*entrystr*) [')] $SP ↑lptr1 [';/EOL] < CH $NP> ↑lptr2
      ) THEN                                                               03078
         FIND SE(*entrystr*) ↑lptr1 ↑lptr2;                                03079
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                            03080
   RETURN (TRUE);                                                          03081
   END.
                                                                           03082
(getiexp) %get EXPAND field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                   0524
   LOCAL TEXT POINTER lptr1, lptr2;                                        0525
   REF entrystr;                                                           0526
   FIND SF(*entrystr*) [')] $NP ↑lptr1 ("Expand"/) ↑lptr2;                 0527
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                            0528
   RETURN (TRUE);                                                          0529
   END.
                                                                           0530
(getimem) %get MEMBERSHIP field from a group/organization ident
entry%
PROCEDURE (entrystr, fldstr, pxr1, ptr2);                                   0531
   LOCAL TEXT POINTER lptr1, lptr2;                                        0532
   REF entrystr;                                                           0533
   IF NOT orgrptst(&entrystr, 0)                                           0534
   OR NOT                                                                  0535
      (FIND SF(*entrystr*) [EOL] $(SP/TAB) ↑lptr1 PT [EOL] < $NP
      ↑lptr2 >) THEN                                                       0536
         FIND SF(*entrystr*) [EOL] ↑lptr1 ↑lptr2;                          0537
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                            0538
   RETURN (TRUE);                                                          0539
   END.
                                                                           0540
(getiprevmem) %get MEMBERSHIPPREVIOUS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                  02541
   LOCAL TEXT POINTER p1, p2;                                              02542
   LOCAL retval;                                                           02543
   REF entrystr;                                                           02544
   retval ← FALSE;                                                         02545
   IF NOT FIND SF(*entrystr*) ["MembershipPrevious:"] $NP ↑p1 [';] <
   CH $NP > ↑p2 THEN                                                       02546
      getiend (&entrystr, $p1, $p2)                                        02547
   ELSE retval ← TRUE;                                                     02548
   stptset (fldstr, ptr1, ptr2, $p1, $p2);                                 02549
   RETURN (retval);                                                        02550
   END.                                                                    02551
                                                                           02552
(geticord) %get COORDINATOR field from a group/organization ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                   0541
   LOCAL TEXT POINTER lptr1, lptr2;                                        0542
   REF entrystr;                                                           0543
   IF NOT orgrptst(&entrystr, 0)                                           0544
```

```
   OR NOT                                                                0545
      (FIND SF(*entrystr*) 3[EOL] $(SP/TAB) ↑lptr1 PT                    0546
         [EOL] <$NP> ↑lptr2 ) THEN                                       0547
            FIND SF(*entrystr*) 3[EOL] ↑lptr1 ↑lptr2;                    0548
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                          0549
   RETURN (TRUE);                                                        0550
   END.
                                                                         0551
(getityp) %get TYPE-OF-ORGANIZATION field from an organization ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                0552
   LOCAL TEXT POINTER lptr1, lptr2;                                      0553
   REF entrystr;                                                         0554
   IF NOT orgrptst(&entrystr, 0)                                         0555
   OR NOT                                                                0556
      (FIND SF(*entrystr*) ["Type:"] $NP ↑lptr1 [';] < CH $NP >
      ↑lptr2 ) THEN                                                      0557
         getiend(&entrystr, $lptr1, $lptr2);                             0558
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                          0559
   RETURN (TRUE);                                                        0560
   END.
                                                                         0561
(getiverify) %get VERIFIED-BY-NIC-PERSONNEL field from an ident
entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                0562
   LOCAL TEXT POINTER lptr1, lptr2;                                      0563
   REF entrystr;                                                         0564
   IF NOT                                                                0565
      (FIND SF(*entrystr*) ["Verified"/"Unverified"] ↑lptr2              0566
         < [EOL] >  NP ↑lptr1) THEN                                      0567
            getiend(&entrystr, $lptr1, $lptr2);                          0568
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                          0569
   RETURN (TRUE);                                                        0570
   END.
                                                                         0571
(getiuser) %get USER (NLS delivery) field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                0572
   LOCAL TEXT POINTER lptr1, lptr2;                                      0573
   LOCAL retval;                                                         0574
   REF entrystr;                                                         0575
   retval ← FALSE;                                                       0576
   IF NOT                                                                0577
      (FIND SF(*entrystr*) ["User:"]                                     0578
         $NP  ↑lptr1 [';] < CH $NP > ↑lptr2) THEN                        0579
            getiend(&entrystr, $lptr1, $lptr2)                           0580
   ELSE                                                                  0581
      retval ← TRUE;                                                     0582
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                          0583
   RETURN(retval);                                                       0584
   END.
                                                                         0585
(getigrps) %get GROUPS field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                                0586
   LOCAL TEXT POINTER lptr1, lptr2;                                      0587
   LOCAL retval;                                                         0588
   REF entrystr;                                                         0589
```

```
   retval ← FALSE;                                                    0590
   IF NOT FIND SF(*entrystr*) ["Groups:"] $NP ↑1ptr1 [';] < CH $NP >
   ↑1ptr2 THEN                                                        0591
      getiend (&entrystr, $1ptr1, $1ptr2)                             0592
   ELSE retval ← TRUE;                                                0593
   stptset (fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                      0594
   RETURN (retval);                                                   0595
   END.                                                               0596
                                                                      0597
(getiphone) %get PHONE field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                             0598
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                   0599
   LOCAL retval;                                                      0600
   REF entrystr;                                                      0601
   retval ← FALSE;                                                    0602
   IF NOT                                                             0603
      (FIND SF(*entrystr*) ["Phone:"] $NP ↑1ptr1 [';] < CH $NP >      0604
         ↑1ptr2) THEN                                                 0605
            getiend(&entrystr, $1ptr1, $1ptr2)                        0606
   ELSE retval ← TRUE;                                                0607
   stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                       0608
   RETURN(retval);                                                    0609
   END.                                                               0610
(getihost) %get NETWORK-DELIVERY HOST field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                             0611
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                   0612
   LOCAL retval;                                                      0613
   REF entrystr;                                                      0614
   retval ← FALSE;                                                    0615
   IF NOT                                                             0616
      (FIND SF(*entrystr*) ["Host:"] $NP ↑1ptr1 [';] < CH $NP >       0617
         ↑1ptr2) THEN                                                 0618
            getiend(&entrystr, $1ptr1, $1ptr2)                        0619
   ELSE retval ← TRUE;                                                0620
   stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                       0621
   RETURN(retval);                                                    0622
   END.                                                               0623
(getinlhost) %get NLS-DELIVERY HOST field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                             0624
   LOCAL TEXT POINTER 1ptr1, 1ptr2;                                   0625
   LOCAL retval;                                                      0626
   REF entrystr;                                                      0627
   retval ← FALSE;                                                    0628
   IF NOT                                                             0629
      (FIND SF(*entrystr*) ["NLS host:"] $NP ↑1ptr1 [';] < CH $NP >
                                                                      0630
         ↑1ptr2) THEN                                                 0631
            getiend#[&entrystr, $1ptr1, $1ptr2)                       0632
   ELSE retval ← TRUE;                                                0633
   stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                       0634
   RETURN(retval);                                                    0635
   END.                                                               0636
(getinma) %get NETWORK MAILBOX Address field from an ident entry%
```

```
PROCEDURE (entrystr  fldstr, ptr1, ptr2);                              0637
   LOCAL TEXT POINTER lptr1, lptr2;                                    0638
   LOCAL retval;                                                       0639
   REF entrystr;                                                       0640
   retval ← FALSE;                                                     0641
   IF NOT                                                              0642
      (FIND SF(*entrystr*) ["Local Network Mailbox Address:"] $NP
      ↑lptr1 [';] NP] < 2CH $NP >                                      0643
         ↑lptr2) THEN                                                  0644
            getiend=[&entrystr, $lptr1, $lptr2)                        0645
   ELSE retval ← TRUE;                                                 0646
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0647
   RETURN(retval);                                                     0648
   END.
                                                                       0649
(getifunction) %get FUNCTION field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0650
   LOCAL TEXT POINTER lptr1, lptr2;                                    0651
   LOCAL retval;                                                       0652
   REF entrystr;                                                       0653
   retval ← FALSE;                                                     0654
   IF NOT                                                              0655
      (FIND SF(*entrystr*) ["Function:"] $NP ↑lptr1 [';] < CH $NP >
                                                                       0656
         ↑lptr2) THEN                                                  0657
            getiend(&entrystr, $lptr1, $lptr2)                         0658
   ELSE retval ← TRUE;                                                 0659
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0660
   RETURN(retval);                                                     0661
   END.
                                                                       0662
(getisorg) %get SECONDARY ORGANIZATION field from an ident entry%
PROCEDURE (entrystr, fldstr  ptr1, ptr2);                              0663
   LOCAL TEXT POINTER lptr1, lptr2;                                    0664
   LOCAL retval;                                                       0665
   REF entrystr;                                                       0666
   retval ← FALSE;                                                     0667
   IF NOT                                                              0668
      (FIND SF(*entrystr*) ["Secondary organization:"] $NP ↑lptr1
      [';] < CH $NP > ↑lptr2 ) THEN                                    0669
         getiend(&entrystr, $lptr1, $lptr2)                            0670
   ELSE retval ← TRUE;                                                 0671
   stptset(fldstr, ptr1, ptr2, $lptr1, $lptr2);                        0672
   RETURN(retval);                                                     0673
   END.
                                                                       0674
(geticapability) %get CAPABILITIES field from an ident entry%
PROCEDURE (entrystr, fldstr, ptr1, ptr2);                              0675
   LOCAL retval;                                                       0676
   LOCAL TEXT POINTER lptr1, lptr2;                                    0677
   REF entrystr;                                                       0678
   retval ← FALSE;                                                     0679
   IF NOT                                                              0680
      (FIND SF(*entrystr*) ["Capabilities:"] $NP ↑lptr1 [';] < CH
      $NP >                                                            0681
         ↑lptr2) THEN                                                  0682
```

```
            getiend(&entrystr, $1ptr1, $1ptr2)                          0683
      ELSE retval ← TRUE;                                               0684
      stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                      0685
      RETURN(retval);                                                   0686
      END.
                                                                        0687
   (getisubcol) %get SUBCOLLECTIONS field from an ident entry%
   PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0688
      LOCAL TEXT POINTER 1ptr1, 1ptr2;                                  0689
      LOCAL retval;                                                     0690
      REF entrystr;                                                     0691
      retval ← FALSE;                                                   0692
      IF NOT                                                            0693
         (FIND SF(*entrystr*) ["Sub-Collection:"] $NP ↑1ptr1 [';] < CH
         $NP >                                                          0694
            ↑1ptr2 ) THEN                                               0695
               getiend(&entrystr, $1ptr1, $1ptr2)                       0696
      ELSE retval ← TRUE;                                               0697
      stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                      0698
      RETURN(retval);                                                   0699
      END.
                                                                        0700
   (getidelivery) %get DELIVERY-TYPES field from an ident entry%
   PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0701
      LOCAL TEXT POINTER 1ptr1, 1ptr2:                                  0702
      LOCAL retval;                                                     0703
      REF entrystr;                                                     0704
      retval ← FALSE;                                                   0705
      IF NOT                                                            0706
         (FIND SF(*entrystr*) ["Delivery:"] $NP ↑1ptr1 [';] < CH $NP >
                                                                        0707
            ↑1ptr2 ) THEN                                               0708
               getiend(&entrystr, $1ptr1, $1ptr2)                       0709
      ELSE retval ← TRUE;                                               0710
      stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                      0711
      RETURN(retval);                                                   0712
      END.
                                                                        0713
   (getimcmnts) %get COMMENTS field from an ident entry%
   PROCEDURE (entrystr, fldstr, ptr1, ptr2);                            0714
      LOCAL TEXT POINTER 1ptr1, 1ptr2;                                  0715
      LOCAL retval;                                                     0716
      REF entrystr;                                                     0717
      retval ← =false:
      if not
         [find sf(*entrystr*) ["Comments:"] $SP ↑1ptr1 [EOL] ↑1ptr2
         ←1ptr2)                                                        0720
            THEN FIND SE(*entrystr*) ↑1ptr1 ↑1ptr2                      0721
      ELSE retval ← TRUE;                                               0722
      stptset(fldstr, ptr1, ptr2, $1ptr1, $1ptr2);                      0723
      RETURN(retval);                                                   0724
      END.
                                                                        0725
%....support routines for get/set entry in ident-file....%              0726
   (getiend) %find the end of an ident entry%
   PROCEDURE (entrystr, ptr1, ptr2);                                    0727
```

```
   REF entrystr, ptr1, ptr2;                                          0728
   IF NOT                                                             0729
      (FIND SF(*entrystr*)                                            0730
         ["Comments:"] EOL < 10 CH > ↑ptr1 ↑ptr2) THEN                0731
            FIND SE(*entrystr*) ↑ptr1 ↑ptr2;                          0732
   RETURN;                                                            0733
   END.
                                                                      0734
(stptset) %set string and/or text-pointers for get routines%
PROCEDURE (string, nptr1, nptr2, optr1, optr2);                       0735
   REF string, nptr1, nptr2, optr1, optr2;                            0736
   IF &string THEN *string* ← optr1 optr2;                            0737
   IF &nptr1 THEN                                                     0738
      FIND optr1 ↑nptr1;                                              0739
   IF &nptr2 THEN                                                     0740
      FIND optr2 ↑nptr2;                                              0741
   RETURN;                                                            0742
   END.
                                                                      0743
(expdtst) %test expand field and set text-pointer to membership
list%
PROCEDURE (string, dstptr);                                           0744
   %This routine looks at the statement indicated by srcptr, and
   returns true or false to indicate whether the expand parameter is
   set.  In addition, if the second argument is non-zero, it assumes
   that this is the address of a t-pointr and updates the pointer to
   point to the beginning of the membership list.  If the membership
   list is not present, then the t-pointer will contain endchr.%      0745
   %--------------------%                                             0746
   REF string, dstptr;                                                0747
   IF FIND SF(*string*) '( $(LD/''/'-) ') $(SP/TAB)                   0748
      "Expand" $(SP/TAB)                                              0749
      ("Group"/"Organization") $(SP/TAB) EOL                          0750
      THEN                                                            0751
         BEGIN %expand parameter set%                                 0752
         IF &dstptr THEN %set dstptr to beginning of membership
         list%                                                        0753
            IF FIND ↑dstptr $(SP/TAB) THEN                            0754
               FIND ↑dstptr;                                          0755
         RETURN(TRUE);                                                0756
         END                                                          0757
   ELSE RETURN(FALSE);                                                0758
   END.
                                                                      0759
%....test type of entry in ident-file....%                            0760
   (jgrptst) %test if a group entry%
   PROCEDURE (string, dstptr);                                        0761
      %This routine looks at the statement indicated by srcptr, and
      returns true or false to indicate whether it contains a group
      identification.  In addition, if the second argument is non-zero,
      it assumes that this is the address of a t-pointr and updates the
      pointer to point to the beginning of the membership list.%      0762
      %--------------------%                                          0763
      REF string, dstptr;                                             0764
      IF FIND SF(*string*) [')] $(SP/TAB) %statement name%            0765
         (("Expand" $(SP/TAB) "Group")/"Group")                       0766
```

```
      $(SP/TAB) EOL THEN                                                0767
         BEGIN %group id present%                                       0768
         IF &dstptr THEN %set dstptr to beginning of membership
         list%                                                          0769
            IF FIND ↑dstptr $(SP/TAB) THEN                              0770
               FIND ↑dstptr;                                            0771
         RETURN(TRUE);                                                  0772
         END                                                            0773
   ELSE RETURN(FALSE);                                                  0774
   END.
                                                                        0775
(orgtst) %test if a organization entry%
PROCEDURE (string, dstptr);                                             0776
   %This routine looks at the statement indicated by srcptr, and
   returns true or false to indicate whether it is an organization.
   In addition, if the second argument is non-zero, it assumes that
   this is the address of a t-pointer and updates the pointer to
   point to the beginning of the membership list.  If the membership
   list is not present, then the t-pointer will contain endchr.%    0777
   %--------------------%                                             0778
   REF string, dstptr;                                                  0779
   IF FIND SF(*string*) [')] $(SP/TAB) %satatement name%                0780
      (("Expand" $(SP/TAB) "Organization")/"Organization")              0781
      $(SP/TAB) EOL  THEN                                               0782
         BEGIN %organization id present%                                0783
         IF &dstptr THEN %set dstptr to beginning of membership
         list%                                                          0784
            IF FIND ↑dstptr $(SP/TAB) THEN                              0785
               FIND ↑dstptr;                                            0786
         RETURN(TRUE);                                                  0787
         END                                                            0788
   ELSE RETURN(FALSE);                                                  0789
   END.
                                                                        0790
(orgrptst) %test if a group or organization entry%
PROCEDURE (string, dstptr);                                             0791
   %This routine looks at string, and returns true or false to
   indicate whether it is an organization.  In addition, if the
   second argument is non-zero, it assumes that this is the address
   of a t-pointer and updates the pointer to point to the beginning
   of the membership list.  If the membership list is not present,
   then the t-pointer will contain endchr.%                             0792
   %--------------------%                                             0793
   LOCAL TEXT POINTER ptr1;                                             0794
   REF string, dstptr;                                                  0795
   IF FIND SF(*string*) [')] $(SP/TAB) ↑ptr1 THEN                       0796
      BEGIN %statement name%                                            0797
      IF FIND ptr1 "Expand" $(SP/TAB) THEN                              0798
         FIND ↑ptr1;                                                    0799
      IF FIND ptr1 ("Group"/"Organization") [EOL] THEN                  0800
         BEGIN %organization/group present%                             0801
         IF &dstptr THEN %set dstptr to beginning of membership
         list%                                                          0802
            IF FIND ↑dstptr $(SP/TAB) THEN                              0803
               FIND ↑dstptr;                                            0804
         RETURN(TRUE);                                                  0805
```

```
        END;                                                    0806
      END;                                                      0807
    RETURN(FALSE);                                              0808
    END.
                                                                0809
%....expand list of idents into list of individuals....%        02841
  (getids) %expand list of idents into list of individuals%
  PROCEDURE (ptr, astr, infotype, idfile);                      02842
    LOCAL expchr, gpstid;                                       02843
    LOCAL TEXT POINTER idf, ide, tmpptr, srcptr, dstptr;        02844
    LOCAL STRING idstr[20], infostr[2000];                      02845
    REF ptr, astr;                                              02846
    %reads an ident from pointer ptr into astring idstr, and then
    calls ckident to get info on it.                            02847
      IF infotype = 0, then it returns all of the info in astr, if
      infotype = 1, then the name only is returned.             02848
      Uses infostr as a work area%                              02849
    expchr ← 0;                                                 02850
    %first, read ident%                                         02851
      LOOP                                                      02852
        BEGIN                                                   02853
        CCPOS ptr;                                              02854
        IF FIND $NP ↑idf '; THEN                                02855
          BEGIN                                                 02856
          IF NOT popids(&ptr) THEN RETURN(FALSE); %no more idents%
                                                                02857
          END                                                   02858
        ELSE EXIT LOOP;                                         02859
        END;                                                    02860
      IF NOT FIND idf / NP / '; / '( /                          02861
        ↑ptr ←ptr ↑ide ←ide THEN                                02862
          err($"Ident List Format Error");                      02863
      FIND ptr ($NP '( /')/ ↑ptr);                              02864
      IF FIND idf ('&/'↑) ↑idf  < CH > THEN expchr ← READC;     02865
      *idstr* ← idf ide; %ident%                                02866
    %Now get info, and check ident%                             02867
      IF ckident($idstr, $infostr, idfile : gpstid) THEN %return
      something%                                                02868
        BEGIN                                                   02869
        IF orgrptst($infostr, 0) THEN                           02870
          BEGIN                                                 02871
          expchr ← CASE expchr OF %expand group list or not%    02872
            ='&: FALSE;                                         02873
            ='↑: TRUE;                                          02874
            ENDCASE expdtst($infostr, 0); %take default from
            ident record%                                       02875
          IF expchr THEN                                        02876
            BEGIN                                               02877
            getimem($infostr, 0,  $dstptr, 0);                  02878
            IF FIND dstptr -EOL %membership list present% THEN
                                                                02879
              BEGIN                                             02880
              pushias(&ptr);                                    02881
              dstptr ← gpstid;                                  02882
              FIND dstptr ↑ptr;                                 02883
              RETURN(getids(&ptr, &astr, infotype, idfile));
```

```
                    END;                                                02884
                  END;                                                  02885
                END;                                                    02886
              END;                                                      02887
            END;                                                        02888
          %Now edit and append to astr %                                02889
            IF infotype = 1 THEN                                        02890
              getifnf($infostr, $infostr);                              02891
            *astr* ← *astr*, *infostr*;                                 02892
      RETURN(TRUE) END.                                                 02893

%....ident pushdown stack support....%                                  02815
    (intids) %initialize ident pushdown stack%
    PROCEDURE (ptr);                                                    02816
      RESET jidstk;                                                     02817
      IF ptr THEN pushids(ptr);                                         02818
      jidsbot ← jidstk;                                                 02819
      RETURN;                                                           02820
      END.                                                              02821

    (popids) %pop the ident pushdown stack%
    PROCEDURE (ptr);                                                    02822
      REF ptr;                                                          02823
      IF jidsbot = jidstk THEN RETURN(FALSE);                           02824
      POP jidstk TO ptr;                                                02825
      RETURN(TRUE);                                                     02826
      END.                                                              02827

    (pushids) %push the ident pushdown stack%
    PROCEDURE (ptr);                                                    02828
      REF ptr;                                                          02829
      PUSH ptr ON jidstk;                                               02830
      RETURN;                                                           02831
      END.                                                              02832

    (gbotids) %collaps ident pushdown stack%
    PROCEDURE (ptr);                                                    02833
      REF ptr;                                                          02834
      IF jidsbot = jidstk THEN RETURN(FALSE);                           02835
      IF jidstk.systks=1 THEN ptr ← [$jidstk+2]                         02836
      ELSE mvbfbf($jidstk+2, &ptr, jidstk.systks);                      02837
      RETURN(TRUE);                                                     02838
      END.                                                              02839
                                                                        02840
%....miscellaneous utility routines....%                                01551
    (getgpids) %given an identlist, return list of only the group
    idents%
    PROCEDURE (ptr, astr, idfnum);                                      02782
      %Read identlist identified by ptr, and return all group idents
      referenced in the string astr, separated by spaces%               02783
      LOCAL TEXT POINTER z1, z2, z3;                                    02784
      LOCAL idfile;                                                     02785
      LOCAL STRING idstr[20], infostr[2000];                            02786
      REF ptr, astr;                                                    02787
      IF NOT idfnum THEN                                                02788
        BEGIN                                                           02789
```

```
      idfile ← 0;                                                    02790
      ON SIGNAL ELSE sigclose(idfile := 0);                          02791
      idfile ← open(0, jfname($"Identfile"));                        02792
      END                                                            02793
   ELSE idfile ← idfnum;                                             02794
   FIND ptr ↑z1;                                                     02795
   WHILE (FIND z1 $NP ('↑/'&/) ↑z2 -'; [ NP / '; / '(] < CH > ↑z3  (
   $NP '( [')] / ) ↑z1 ) DO                                          02796
      BEGIN                                                          02797
      *idstr* ← z2 z3;                                               02798
      IF NOT ckident($idstr, $infostr, idfile) THEN
      err($"Identification System Error");                           02799
      IF orgrptst($infostr, 0) THEN *astr* ← *astr*, SP, *idstr*;
                                                                     02800
      END;                                                           02801
   IF NOT idfnum THEN close(idfile := 0);                            02802
   RETURN;                                                           02803
   END.                                                              02804
(stnamcap) %Capitalize first letter of each word in a string%
PROCEDURE (string);                                                  01552
   LOCAL char, count;                                                01553
   REF string;                                                       01554
   %----------------%                                                01555
   count ← 1;                                                        01556
   CASE (char ← *string*[count]) OF                                  01557
      IN ['a, 'z]:  *string*[count] ← char - 40B;                    01558
      =SP:                                                           01559
         BEGIN                                                       01560
         *string*. ← *string*[count+1 TO string.L];                  01561
         REPEAT CASE;                                                01562
         END;                                                        01563
      ENDCASE;                                                       01564
   WHILE (count ← count + 1) < string.L                              01565
      DO                                                             01566
      IF *string*[count] = SP                                        01567
      AND (char ← *string*[count + 1]) IN ['a, 'z] THEN              01568
         BEGIN                                                       01569
         *string*[count + 1] ← char - 40B;                           01570
         count ← count + 1;                                          01571
         END;                                                        01572
   RETURN;                                                           01573
   END.                                                              01574

(namesearch) %search and print routine%
PROCEDURE (string, idstr, type, fileno);                             03083
   %This procedure accepts a string containing a last name in
   namstr, and searches the identification file for entries with
   matching string -- last name, first character(s) of last name for
   individuals, or string in name...depending on type. When a match
   is found information is typed to the user, which is intended to
   identify the entry.  The number of hits is returned, plus if the
   number is greater than 0, idstr has the ident of the last one
   processed.%                                                       03084
                                                                     03085
   LOCAL foundsome, retvalue,  savrubabt, stid, typid, idstid, i;
                                                                     03086
```

```
LOCAL STRING work[1000], tempsr[200], uppercasestring[200];  03087
REF idstr, string;                                           03088
%Open id file%                                               03089
   idstid ← orgstid;                                         03090
   IF NOT fileno THEN                                        03091
      BEGIN                                                  03092
      ON SIGNAL ELSE                                         03093
         sigclose(idstid.stfile := 0);                       03094
      idstid.stfile ← open(0, jfname($"identfile"));         03095
      END                                                    03096
   ELSE idstid.stfile ← fileno;                              03097
savrubabt ← rubabt := FALSE;                                 03098
retvalue ← 0;                                                03099
CASE type OF                                                 03100
   = lname: %individual's last names%                        03101
      BEGIN                                                  03102
      FOR i ← 1 UP UNTIL > string.L DO %remove leading spaces%
                                                             03103
         IF *string*[i] # SP THEN                            03104
            BEGIN                                            03105
            IF i NOT= 1 THEN                                 03106
               *string* ← *string*[i TO string.L];           03107
            EXIT;                                            03110
            END;                                             03109
      FOR i ← string.L DOWN UNTIL <= 0 DO %remove trailing
      spaces%                                                03110
         IF *string*[i] # SP THEN                            03111
            BEGIN                                            03112
            IF i NOT= string.L THEN                          03113
               *string* ← *string*[1 TO i];                  03114
            EXIT;                                            03115
            END;                                             03116
      *tempsr* ← '', *string*, '';                           03117
      FOR i ← 1 UP UNTIL > tempsr.L DO %convert spaces to ''%
                                                             03118
         IF *tempsr*[i] = SP THEN *tempsr*[i] ← '';          03119
      IF (stid ← namelook(idstid, $tempsr)) = endfil OR NOT (FIND
      SF(stid) [')] $SP "LAST NAME") THEN                    03120
         BEGIN                                               03121
         fbctl(fbaddlit, $"
         None
         ");                                                 03122
         EXIT CASE;                                          03123
         END;                                                03124
      fbctl(fbaddlit, $"
      The following individuals with last name ");           03125
      fbctl(fbaddlit, &string);                              03126
      fbctl(fbaddlit, $" are already defined
      ");                                                    03127
      retvalue ← idpiname(stid, &idstr);                     03128
      END;                                                   03129
   = idchr:                                                  03130
      BEGIN                                                  03131
      IF (stid ← namelook(idstid, $"'individuals'")) = endfil OR
      (stid := getsub(stid)) = stid THEN EXIT CASE;          03132
      fbctl(fbaddlit, $"
```

```
   The following individuals with last names begining with the
   letter(s) ");                                                   03133
   fbctl(fbaddlit, &string);                                       03134
   fbctl(fbaddlit, $" are already defined
   ");                                                             03135
   FOR i ← 1 UP UNTIL > string.L DO %remove leading spaces%
                                                                   03136
      IF *string*[i] # SP THEN                                     03137
         BEGIN                                                     03138
         IF i NOT= 1 THEN                                          03139
            *string* ← *string*[i TO string.L];                    03140
         EXIT;                                                     03141
         END;                                                      03142
   *tempsr* ← *string*;                                            03143
   FOR i ← 1 UP UNTIL > tempsr.L DO %convert spaces to ''%
                                                                   03144
      IF *tempsr*[i] = SP THEN *tempsr*[i] ← '';                   03145
   LOOP                                                            03146
      BEGIN                                                        03147
      IF FIND SF(stid) $SP '( $SP  '' *tempsr* [')] $SP "LAST
      NAME" THEN                                                   03148
         retvalue ← retvalue + idplname(stid, &idstr);             03149
      IF getftl(sxid)  OR inptrf := FALSE THEN EXIT LOOP;          03150
      stid ← getsuc(stid);                                         03151
      END;                                                         03152
   IF NOT retvalue THEN                                            03153
      fbctl(fbaddlit, $"
      None
      ");                                                          03154
   END;                                                            03155
= strinname: %scan names for lit%                                  03156
   BEGIN                                                           03157
   *uppercasestring* ← *string*;                                   03158
   astruc($uppercasestring);                                       03159
   %process individuals%                                           03160
      IF (stid ← namelook(idstid,  "'individuals'")) NOT=
      endfil AND (stid := getsub(stid)) NOT= stid THEN             03161
         BEGIN                                                     03162
         fbctl(fbaddlit, $"
         Following is a list of individuals which have ");
                                                                   03163
         fbctl(fbaddlit, &string);                                 03164
         fbctl(fbaddlit, $" in their names:
         ");                                                       03165
         foundsome ← 0;                                            03166
         LOOP                                                      03167
            BEGIN                                                  03168
            IF (FIND SF(stid)  [')] $SP "LAST NAME") AND (stid
            := getsub(stid)) NOT= stid THEN                        03169
               LOOP                                                03170
                  BEGIN                                            03171
                  *work* ← SF(stid) SE(stid);                      03172
                  getifnf($work, $tempsr,0,0); %first name
                  first%                                           03173
                  astruc($tempsr);                                 03174
                  IF FIND SF(*tempsr*) [*uppercasestring*]
```

```
                    THEN                                                03175
                       BEGIN                                            03176
                       idpind($work, & dstr);                           03177
                       BUMP foundsome;                                  03178
                       crlf();                                          03179
                       END;                                             03180
                    IF inptrf := FALSE THEN EXIT LOOP 2;                03181
                    IF getftl(stid) THEN                                03182
                       BEGIN                                            03183
                       stid ← getsuc(stid);                             03184
                       EXIT LOOP;                                       03185
                       END;                                             03186
                    stid ← getsuc(stid);                                03187
                    END;                                                03188
              IF getftl(stid) OR inptrf := FALSE THEN EXIT LOOP;
                                                                        03189
              stid ← getsuc(stid);                                      03190
              END;                                                      03191
           IF NOT foundsome THEN                                        03192
              fbctl(fbaddlit, $"
              None
              ");                                                       03193
           retvalue ← retvalue + foundsome;                             03194
           END;                                                         03195
        %process groups%                                                03196
           IF (stid ← namelook(idstid, $"'groups'")) NOT= endfil
           AND (stid := getsub(stid)) NOT= stid THEN                    03197
              BEGIN                                                     03198
              fbctl(fbaddlit, $"
              Following is a list of groups which have ");              03199
              fbctl(fbaddlit, &string);                                 03200
              fbctl(fbaddlit, $" in their names:
              ");                                                       03201
              foundsome ← 0;                                            03202
              LOOP                                                      03203
                 BEGIN                                                  03204
                 *work* ← SF(stid) SE(stid);                            03205
                 getinam($work, $tempsr,0,0);                           03206
                 astruc($tempsr);                                       03207
                 IF FIND SF(*tempsr*) /*uppercasestring*/ THEN          03208
                    BEGIN                                               03209
                    idpgrporg($work, &idstr);                           03210
                    BUMP foundsome;                                     03211
                    crlf();                                             03212
                    END;                                                03213
                 IF getftl(stid) OR inptrf := FALSE THEN EXIT LOOP;
                                                                        03214
                 stid ← getsuc(stid);                                   03215
                 END;                                                   03216
              IF NOT foundsome THEN                                     03217
                 fbctl(fbaddlit, $"
                 None
                 ");                                                    03218
              retvalue ← retvalue + foundsome;                          03219
              END;                                                      03220
        %process organizations%                                         03221
```

```
            IF (stid ← namelook(idstid, $"'organizations'")) NOT=
            endfil AND (stid := getsub(stid)) NOT= stid THEN     03222
               BEGIN                                             03223
               fbctl(fbaddlit, $"
               Following is a list of organizations which have ");
                                                                 03224
               fbctl(fbaddlit, &string);                         03225
               fbctl(fbaddlit, $" in their names:
               ");                                               03226
               foundsome ← 0;                                    03227
               LOOP                                              03228
                  BEGIN                                          03229
                  *work* ← SF(stid) SE(stid);                    03230
                  getinam($work, $tempsr,0,0);                   03231
                  astruc($tempsr);                               03232
                  IF FIND SF(*tempsr*) [*uppercasestring*] THEN  03233
                     BEGIN                                       03234
                     idpgrporg($work, &idstr);                   03235
                     BUMP foundsome;                             03236
                     crlf();                                     03237
                     END;                                        03238
                  IF getftl(stid)  OR inptrf := FALSE THEN EXIT
                  LOOP;                                          03239
                  stid ← getsuc(stid);                           03240
                  END;                                           03241
               IF NOT foundsome THEN                             03242
                  fbctl(fbaddlit, $"
                  None
                  ");                                            03243
               retvalue ← retvalue + foundsome;                  03244
               END;                                              03245
         END;                                                    03246
      ENDCASE;                                                   03247
   rubabt ← savrubabt;                                           03248
   IF NOT fileno THEN close(idstid.stfile := 0);                 03249
   RETURN(retvalue);                                             03250
   END.
                                                                 03251

(idplname) %print a last name branch of individuals%
PROCEDURE (stid, idstr);                                         01746
   LOCAL count;                                                  01747
   LOCAL STRING work[2000];                                      01748
   REF idstr;                                                    01749
   count ← 0;                                                    01750
   IF (stid := getsub(stid)) = stid THEN RETURN(count);          01751
   LOOP                                                          01752
      BEGIN                                                      01753
      BUMP count;                                                01754
      *work* ← SF(stid) SE(stid);                                01755
      idpind($work, &idstr);                                     01756
      IF getftl(stid) OR inptrf := FALSE THEN EXIT LOOP;         01758
      stid ← getsuc(stid);                                       01759
      END;                                                       01760
   RETURN(count);                                                01761
   END.                                                          01762
                                                                 01763
```

```
(idpind) %format and print abreviated individual's ident entry%
PROCEDURE (string, idstr);                                          01764
   LOCAL STRING work[250];                                          01765
   LOCAL TEXT POINTER  orgf, orge, namef, namee;                    01766
   REF idstr;                                                       01767
   getiid(string, &idstr, 0,0);                                     01768
   getiorg(string, 0, $orgf, $orge);                                01769
   getinam(string, 0, $namef, $namee);                              01770
   *work* ← namef namee, ", Organization: ", orgf orge, ", Ident =
   ", *idstr*, EOL;                                                 01771
   fbctl(fbaddlit, $work);                                          01772
   RETURN;                                                          01773
   END.                                                             01774
                                                                    01775
(idpgrporg) %format abbreviated group or organization entry%
PROCEDURE (string, idstr);                                          01776
   LOCAL STRING work[250];                                          01777
   LOCAL TEXT POINTER namef, namee;                                 01778
   REF idstr;                                                       01779
   getiid(string, &idstr, 0,0);                                     01780
   getinam( STRING, 0, $namef, $namee);                              01781
   *work* ← namef namee, ", Ident = ", *idstr*, EOL;                01782
   fbctl(fbaddlit, $work);                                          01783
   RETURN;                                                          01784
   END.                                                             01785
                                                                    01786
(idflsearch) %search ident-file for last name or content-in-name%
PROCEDURE (type, idstr, fileno);                                    01787
   %Searches id file for individual's last name, first characters of
   individual's last names, or string, depending on type.  Returns
   true or false depending on whether an ident is accepted.. If
   true, idstr contains the ident.%                                 01788
   %------------%                                                  01789
   LOCAL  count, idstid, lnamstr;                                   01790
   LOCAL STRING newidstr[20], work[500];                            01791
   LOCAL TEXT POINTER tptr1, tptr2;                                 01792
   REF idstr;                                                       01793
   idstid ← orgstid;                                                01794
   IF fileno THEN                                                   01795
      idstid.stfile ← fileno                                        01796
   ELSE                                                             01797
      BEGIN                                                         01798
      ON SIGNAL ELSE sigclose(idstid.stfile := 0);                  01799
      idstid.stfile ← open(0, jflname($"identfile"));               01800
      END;                                                          01801
   CASE namesearch(&idstr, $newidstr, type, idstid.stfile) OF       01802
      = 1 : %only one hit -- is it correct%                         02744
         BEGIN                                                      01803
         IF type = lname THEN                                       01804
            BEGIN                                                   01805
            fbctl(fbaddlit, $"
            Is this the correct ");                                 01806
            fbctl(fbaddlit, &idstr);                                01807
            fbctl(fbaddlit, $"?");                                  01808
            END                                                     01809
         ELSE fbctl(fbaddlit, $"
```

perfuring etc

```
         Is this the correct one? ");                                01810
         IF oldanswer() THEN                                         01811
            BEGIN                                                    01812
            *idstr* ← *newidstr*;                                    01814
            fbctl(fbaddlit, $"
            Ident ");                                                01815
            fbctl(fbaddlit, &idstr);                                 01816
            fbctl(fbaddlit, $" accepted
            ");                                                      01817
            IF NOT fileno THEN close(idstid.stfile := 0);            02771
            RETURN(TRUE);                                            01824
            END;                                                     01825
         END;                                                        01827
      > 1 : %more than one hit -- ask for correct one%               02745
         BEGIN                                                       02746
         fbctl(fbaddlit, $"
         Type the correct IDENT: ");                                 02749
         *idstr* ← NULL;                                             02772
         rdlit(&idstr, 0);                                           02765
         IF idstr.L THEN                                             02766
            BEGIN                                                    02768
            IF NOT fileno THEN close(idstid.stfile := 0);            02770
            RETURN(TRUE);                                            02767
            END;                                                     02769
         END;                                                        02763
      ENDCASE                                                        02764
         BEGIN                                                       02805
         *idstr* ← NULL;                                             02807
         END;                                                        02814
   curchr ← 'N;                                                      01828
   IF NOT fileno THEN close(idstid.stfile := 0);                     01829
   RETURN(FALSE);                                                    01830
   END.
                                                                     01831
(makgid) %generate an ident from group name string%
PROCEDURE (namestr, idstr);                                          01832
                                                                     01833
   LOCAL count;                                                      01834
   LOCAL TEXT POINTER tptr1, tptr2;                                  01835
   REF namestr, idstr;                                               01836
   count ← empty + 1;                                                01837
   *idstr* ← *namestr*[count];                                       01838
   FIND SF(*namestr*) ↑tptr1 ↑tptr2;                                 01839
   WHILE (count ← count+1) <= idstr.M DO                             01840
      IF (FIND tptr1 [SP] [L] ↑tptr1 ←tptr1 ↑ tptr2) THEN             01841
         *idstr* ← *idstr*, tptr1 tptr2                              01842
      ELSE EXIT LOOP;                                                01843
   RETURN;                                                           01844
   END.
                                                                     01845
(ckilmem) %verify logical membership of an ident in an ident list%
PROCEDURE (ident, l1, l2, openidfileno);                             02411
   %----------%                                                      02412
   LOCAL outcome, idfileno;                                          02413
   LOCAL STRING grpslist [500], grpsident [35], identry [1000];      02414
   LOCAL TEXT POINTER p1, p2, g1, g2;                                02415
```

```
REF ident, l1, l2, lgngrps;                                          02416
%----------%                                                         02417
idfileno ← openidfileno;                                             02418
%null list?%                                                         02419
   IF NOT (FIND l1 > (LD/'-) ↑p1) OR p1 [1] > l2 [1] THEN            02420
      RETURN (FALSE);                                                02421
%user appear explicitly in list?%                                    02422
   IF ckipmem (&ident, &l1, &l2) THEN RETURN (TRUE);                 02423
%locate ident's group list%                                          02424
   IF *ident* = *initsr* AND &lgngrps THEN                           02425
      FIND SF(*lgngrps*) ↑g1 SE(*lgngrps*) ↑g2                       02426
   ELSE                                                              02427
      BEGIN                                                          02428
      IF NOT idfileno THEN                                           02429
         BEGIN                                                       02430
         idfileno ← open (0, jflname ($"Identfile"));                02431
         ON SIGNAL ELSE IF idfileno AND NOT openidfileno THEN
                                                                     02432
            sigclose (idfileno := 0);                                02433
         END;                                                        02434
      IF NOT ckident (&ident, $identry, idfileno) THEN err (0);
                                                                     02435
      getigrps ($identry, $grpslist, 0, 0);                          02436
      FIND SF(*grpslist*) ↑g1 SE(*grpslist*) ↑g2;                    02437
      END;                                                           02438
outcome ← FALSE;                                                     02439
IF g1 [1] < g2 [1] THEN                                              02440
   BEGIN                                                             02441
   %list contain a group of which the ident is a member?%            02442
      FIND g1 ↑p2;                                                   02443
      WHILE (FIND p2 > $(SP/',) ↑p1 1$(LD/'-) ↑p2) DO                02444
         BEGIN                                                       02445
         *grpsident* ← p1 p2;                                        02446
         IF ckipmem ($grpsident, &l1, &l2) THEN                      02447
            BEGIN                                                    02448
            outcome ← TRUE;                                          02449
            GOTO ckilexit;                                           02450
            END;                                                     02451
         END;                                                        02452
   %how about a group of which the ident is INDIRECTLY a member?%
                                                                     02453
      IF NOT idfileno THEN                                           02454
         BEGIN                                                       02455
         idfileno ← open (0, jflname ($"Identfile"));                02456
         ON SIGNAL ELSE IF idfileno AND NOT openidfileno THEN
                                                                     02457
            sigclose (idfileno := 0);                                02458
         END;                                                        02459
      FIND g1 ↑p2;                                                   02460
      WHILE (FIND p2 > $(SP/',) ↑p1 1$(LD/'-) ↑p2) DO                02461
         BEGIN                                                       02462
         *grpsident* ← p1 p2;                                        02463
         IF ckilmem ($grpsident, &l1, &l2, idfileno) THEN            02464
            BEGIN                                                    02465
            outcome ← TRUE;                                          02466
            GOTO ckilexit;                                           02467
```

```
            END;                                                    02468
         END;                                                       02469
      END;                                                          02470
   (ckilexit): IF idfileno AND NOT openidfileno THEN                02471
      close (idfileno := 0);                                        02472
   RETURN (outcome);                                                02473
   END.                                                             02474
                                                                    02475
(ckipmem) %verify physical membership of an ident in an ident list%
PROCEDURE (ident, l1, l2);                                          02476
   %----------%                                                     02477
   LOCAL STRING listident [35];                                     02478
   LOCAL TEXT POINTER p1, p2;                                       02479
   REF ident, l1, l2;                                               02480
   %----------%                                                     02481
   FIND l1 >;                                                       02482
   WHILE ((FIND $(SP/',) ↑p1 1$(LD/'-) ↑p2) AND (p2 [1] <= l2 [1]))
   DO                                                               02483
      BEGIN                                                         02484
      *listident* ← p1 p2;                                          02485
      IF *ident* = *listident* THEN RETURN (TRUE);                  02486
      END;                                                          02487
   RETURN (FALSE);                                                  02488
   END.                                                             02489
                                                                    02490
(clnidlst) %clean ident list -- of comments, expansion chars, etc.%
PROCEDURE (inlist, %appends to->% outlist);                         02491
   %----------%                                                     02492
   LOCAL STRING ident [40];                                         02493
   LOCAL TEXT POINTER l1, l2, p1, p2, p3;                           02494
   REF inlist, outlist;                                             02495
   %----------%                                                     02496
   FIND SF(*outlist*) ↑l1;                                          02497
   IF outlist.L AND *outlist* [outlist.L] # SP THEN                 02498
      *outlist* ← *outlist*, SP;                                    02499
   FIND SF(*inlist*) ↑p3;                                           02500
   WHILE (FIND p3 > $(SP/',) ('&/'↑/) ↑p1 1$(LD/'-) ↑p2 ($(SP/',) '(
   [')] /) ↑p3) DO                                                  02501
      BEGIN                                                         02502
      *ident* ← p1 p2;                                              02503
      FIND SE(*outlist*) ↑l2;                                       02504
      IF NOT ckipmem ($ident, $?L1, $l2) THEN                        02505
         *outlist* ← *outlist*, *ident*, SP;                        02506
      END;                                                          02507
   IF *outlist* [outlist.L] = SP THEN BUMP DOWN outlist.L;          02508
   RETURN;                                                          02509
   END.                                                             02510
                                                                    02511
(delcmidents) %delete idents common to both ident lists%
PROCEDURE (lhlist, lhdelete, rhlist, rhdelete);                     02512
   %----------%                                                     02513
   LOCAL STRING lhident [40], rhident [40];                         02514
   LOCAL TEXT POINTER lh1, lh2, lht, rh1, rh2, rht;                 02515
   REF lhlist, rhlist;                                              02516
   %----------%                                                     02517
   FIND SE(*lhlist*) (';/) ↑lh1;                                    02518
```

```
    WHILE (FIND lh1 < ↑lht $(SP/',) ↑lh2 1$(LD/'-) ↑lh1) DO                02519
       BEGIN                                                               02520
       *lhident* ← lh1 lh2;                                                02521
       FIND SE(*rhlist*) (';/) ↑rh1;                                       02522
       WHILE (FIND rh1 < ↑rht $(SP/',) ↑rh2 1$(LD/'-) ↑rh1) DO             02523
          BEGIN                                                            02524
          *rhident* ← rh1 rh2;                                             02525
          IF *lhident* = *rhident* THEN                                    02526
             BEGIN                                                         02527
             IF lhdelete THEN ST lh1 lht ← NULL;                           02528
             IF rhdelete THEN ST rh1 rht ← NULL;                           02529
             EXIT LOOP;                                                    02530
             END;                                                          02531
          END;                                                             02532
       END;                                                                02533
    IF lhdelete AND FIND SE(*lhlist*) (';/) ↑lh2 1$(SP/',) ↑lh1 THEN
                                                                           02534
       ST lh1 lh2 ← NULL;                                                  02535
    IF rhdelete AND FIND SE(*rhlist*) (';/) ↑rh2 1$(SP/',) ↑rh1 THEN
                                                                           02536
       ST rh1 rh2 ← NULL;                                                  02537
    RETURN;                                                                02538
    END.                                                                   02539
                                                                           02540
FINISH

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

                                                                           02393
                  where all of the idents have the same lastname (the
                  name of the head of the branch).                         02394
               .
               .
               .                                                           02395
            .
            .
            .                                                              02396
         ('groups')                                                        02397
            (ident) Expand Group
            membershiplist
```

```
        groupname
        cordinatorIdent
        Address

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

                                                                    02401
        .
        .
        .                                                           02402
%                                                                   02403
```