PARSER.

```
< NLS, PARSER.NLS;74, >, 4-OCT-74 22:50 CHI ;;;;
FILE parser % L10 <rel-nls>parser.rel %% (L10,) (rel-nls,parser.rel,) %
                                                                    02
% DOCUMENTATION %                                                   03
   % control environment for the interpreter %                      04
      % the interpreter is controlled by two processing stacks: the
      "path stack" and the "eval stack".  The path stack records the
      path through the grammar and contains values retruned from
      processing funtions.  The eval stack holds pointers to path stack
      records corresponding to the dynamic evaluation of the parser.
      The interpreter is a stack machine, and values for function
      actual parameters and pointers to the return value records are
      contained in the eval stack %                                 05
      % the functions recognized by the interpreter cause pointers to
      path stack records to be popped and pushed onto the eval stack %
                                                                    06
   % the use of interpreter variables %                            07
      % interpreter variables are data cells which contain pointers to
      path stack records.  A check is made whenever a variable is
      referenced that the pointer is still valid %                  08
% DECLARATIONS %                                                    09
   % REGISTERS %                                                   010
      REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, p = 7, m = 10, s = 9; 011
   % REF VARIABLES %                                             0165
      REF fbstr, inpt;                                           0166
      REF tda, sysmsg, nlpcmdstk;                                0187
% EXECUTION CONTROL ROUTINES %                                   0188
   (initsubsystems) PROC; %* defines the "normal" subsystems available
   from NLS %                                                   05869
      % RETURNS %                                               05870
         % none %                                               05871
      LOCAL % VARIBLES %                                        05872
         numwsubsys,    %number or word for each subsystem/program name%
                                                                05873
         subptr,        %ptr to string containing subsystem/program
         name%                                                  05874
         sdptr,         %ptr to subsystem dispatch record%     05875
         notlevel1,     % ctrl bits for not level 1 name %      05876
         ctrlbits;      % interpreter contol bits for keyword
         recognition %                                          05877
      REF subptr;                                               05878
      % ---------------- %                                      05879
      % NOTE *** the first subsystem defined here becomes the "base
      level" or default subsystem for NLS %                     05880
      ctrlbits ← 7B; % Level 1 + DNLS + TNLS %                  05881
      notlevel1 ← 3B; % DNLS + TNLS %                           05882
      % define all subsystems %                                 05883
         dfnsubsys( $nlseditor, ctrlbits ,$allsubs);         %
         nlseditor %                                            05884
         dfnsubsys( $subcalculator, ctrlbits ,$allsubs);     %
         subcalculator %                                        05885
         %dfnsubsys( $subident, ctrlbits ,$allsubs);%        %
         subident %                                             05887
         %                                                      05888
         dfnsubsys( $subreadmail, ctrlbits ,$allsubs);
         %% read mail %%                                       05889
```

```
        %                                                              05890
        dfnsubsys( $subsendmail, ctrlbits ,$allsubs);
        % send mail %                                                  05891
        dfnsubsys( $subsyntax, notlevel1 ,$allsubs);          % syntax
        generator %                                                    06557
        dfnsubsys( $subprograms, ctrlbits ,$allsubs);
        % user subprograms %                                          05892
        dfnsubsys( $subuser, ctrlbits ,$allsubs);             %
        user-options %                                                 05893
        dfnsubsys( $subxxx, ctrlbits ,$allsubs);              %
        systems personnel subsystem %                                  05922
        dfnsubsys( $subsupervisor, notlevel1 ,$allsubs);
        % user-options %                                               05894
    %define the subsystems specified in user option page%             05895
        %check to see if this user's options are set up%               05896
        numwsubsys ← ((usys1.M + 4) / 5) + 1;                          05898
        FOR &subptr ← $usys2 UP numwsubsys UNTIL > $usys15 DO          05899
            IF subptr.L THEN                                           05900
                IF sdptr ← getsdptr(&subptr,$allsubs) THEN             05901
                    dfnsubsys(sdptr,gtctrlbits(sdptr),$nlssubs)        05919
                ELSE                                                   05902
                    BEGIN                                              05906
                    ON SIGNAL                                          05907
                    ELSE                                               05908
                        BEGIN                                          05910
                        ON SIGNAL ELSE;                                05921
                        dismes(2,&sysmsg);                             05909
                        REPEAT LOOP;                                   05911
                        END;                                           05912
                    getuprog( &subptr);                                05905
                    ON SIGNAL ELSE;                                    05920
                    END;                                               05913
    RETURN;                                                            05903
    END.
                                                                       05904
(getsdptr)  %gets a ptr to s subsystem dispatch record%                05646
    PROCEDURE (                                                        05647
    % FORMAL ARGUEMENTS%                                               05648
        stptr, %ptr to a string containing subsystem name%             05649
        subrule);     %ptr to a subsystem rule  nlssubs or allsubs%
                                                                       05650
    % RETURNS %                                                       05651
        % ptr to the appropiate subsystem dispatch record  or FALSE %
                                                                       05652
    LOCAL   %Variables%                                                05653
        sdptr,          %ptr to the appropiate subsystem dispatch
        record%                                                        05654
        frameptr;       %ptr to current alternative of the rule subrule%
                                                                       05655
    REF %Variables%                                                    05656
        stptr, %ptr to a string containing subsystem name%             05657
        subrule,                                                       05658
        frameptr;                                                      05659
    % set up frameptr and check to see we have really been passed a
    valid rule%                                                        05660
        &frameptr ← &subrule;                                          05661
```

```
      IF frameptr.opcode NOT= $xecute THEN RETURN(FALSE);          05662
      &frameptr ← frameptr.addr;                                   05663
      WHILE &frameptr DO                                           05664
         IF */frameptr.addr/* = *stptr* THEN                       05665
            BEGIN                                                  05666
            % the ptr to the subsystem dispatch record is the
            address field of the sucessorof this frame%            05667
            &frameptr ← frameptr.nsuccessor;                       05668
            sdptr ← frameptr.addr;                                 05669
            RETURN(sdptr)                                          05670
            END                                                    05671
         ELSE &frameptr ← frameptr.alternative;                    05672
      RETURN(FALSE);                                               05673
   END.                                                            05674
(getuprog) PROCEDURE                                               05675
   %FORMAL PARAMETERS%                                             05676
      (stptr);       %ptr to a string containg program name (or a
      directory name followed by a comma follwed by the file name.)%
                                                                   05677
   % This procedure calls gpget with a flag saying not to put out
   program loading messages except for errors. %                   06333
   LOCAL                                                           05678
      adstr[40];                                                   05679
   LOCAL TEXT POINTER tptl;                                        05680
   LOCAL STRING locstr[50];                                        05681
   REF stptr;                                                      05682
   % set up local string and text pointer%                         05683
      *locstr* ← '< , *stptr* , ",>";                              05684
      FIND SF(*locstr*) ↑tptl;                                     05685
      lnkprs($tptl,$adstr);                                        05686
      RETURN(gpget($adstr, FALSE % don't print out messages %));
                                                                   05687
   END.                                                            05688
(gotohelp) PROCEDURE;                                              06436
   % save the accumulators %                                       06437
   svacl ← rl; rl ← $svacs; !BLT rl, svacse;                       06438
   s ← s + 400000408;                                              06439
   enthelp( FALSE);                                                06440
   !HRLZI rl, svacs;                                               06441
   !BLT rl, 17B;                                                   06442
   rl ← svacl;                                                     06443
   !JSYS debrk;                                                    06444
   END.
                                                                   06445
(enthelp) PROCEDURE ( mode % If TRUE we are in help already % ); %
set up to $enter the subhelp system after a ↑Q %                   06558
   LOCAL % VARIABLES %                                             06559
      subsysptrloc,          % dummy params for $call to xgoto %   06560
      dispatchptrloc, % ditto %                                    06561
      exflagptrloc,          % ditto %                            06562
      entryptr,      % ptr to current subsystem stack entry %     06563
      nextptr,       % ptr to bottom of subsystem stack %         06564
      curptr;        % ptr to path stack entry %                  06565
   REF entryptr, nextptr, curptr;                                  06566
   % hlpcmdstk REFd at start of file %                             06567
   % this routine sets up the global string pointers hlpcmdstk to
```

```
point to the place into the place in the data base to which the
user is taken.  It then fakes an EXECUTE HELP so that the user
will end up in the subhelp system.  Note that we do not go
directly there, but end up there when the next input character
was read %                                                          06568
%------------%                                                      06569
% set up ptr to subsystem name %                                    06570
   &entryptr ← $sbstack + sbstkx - $spentsize;                      06571
IF NOT mode THEN                                                    06572
   BEGIN                                                            06573
   % Get storage for name stack %                                    06574
      IF NOT &hlpcmdstk THEN &hlpcmdstk ← getarray(hpcmdmax + 1,
      $dspblk);                                                     06575
   %  set ptr to current command (if there is one) %                06576
      &curptr ← $pathstk;                                           06577
      hlpcmdstk ← 1;                                                06578
      hlpcmdstk[1] ← entryptr.spnptr;                               06579
      IF pathx > 0 % not at command reset state % THEN              06580
         BEGIN                                                      06581
         &nextptr ← $pathstk + pathx;                               06582
         % get the rest of the path %                               06583
         DO                                                         06584
            BEGIN                                                   06585
            CASE curptr.pfuncton OF                                 06586
               = $keywrec:                                          06587
                  BEGIN                                             06588
                  BUMP hlpcmdstk;                                   06589
                  hlpcmdstk[hlpcmdstk] ←
                  [curptr.curnodeptr].addr;                         06590
                  END;                                              06591
               = $xselect: NULL;                                    06592
                                                                    06593
                  % BEGIN                                           06594
                  hlpcmdstk ← 2;                                    06595
                  hlpcmdstk[1] ← $"operands";                       06596
                  hlpcmdstk[2] ←                                    06597
                     CASE [curptr.curnodeptr].opcode OF             06598
                        = $ssel: $"source";                         06599
                        = $dsel: $"destination";                    06600
                        = $lsel: $"content";                        06601
                        ENDCASE $"source";                          06602
                  END; %                                            06603
                                                                    06604
               = $xviewspecs:                                       06605
                  BEGIN                                             06606
                  hlpcmdstk ← 2;                                    06607
                  hlpcmdstk[1] ← $"operands";                       06608
                  hlpcmdstk[2] ← $"viewspecs";                      06609
                  END;                                              06610
               = $xlevadj:                                          06611
                  BEGIN                                             06612
                  hlpcmdstk ← 2;                                    06613
                  hlpcmdstk[1] ← $"operands";                       06614
                  hlpcmdstk[2] ← $"level-adjust";                   06615
                  END;                                              06616
               = $xconfirm:                                         06617
```

```
                    BEGIN                                                06618
                    hlpcmdstk ← 1;                                       06619
                    hlpcmdstk[1] ← $"ok";                                06620
                    END;                                                 06621
                ENDCASE;                                                 06622
              &curptr ← &curptr + $totalrecsize;                         06623
              END                                                        06624
            UNTIL (&curptr >= &nextptr) OR (hlpcmdstk = hpcmdmax);
                                                                         06625
            END;                                                         06626
      END;                                                               06627
   % set up to execute the help rule (with the current subsystem
   name visible) %                                                       06628
      % Fake dispatch record: current subsystem name, apprpriate
      help rule. %                                                       06629
         subsysptrloc ← helploc.dptname ← entryptr.sbnptr; %
         subsystem name %                                                06630
         helploc.dptrun ←  IF nlmode = fulldisplay THEN $qdhelp          06631
            ELSE $qthelp; % rule to be executed %                        06632
         helploc.dptvalid ← $dptvldationcode;                            06633
         helploc.dptinit ← helploc.dptfinish ←  helploc.dptnotused ←
         helploc.dptrentry ← 0;                                          06634
      dispatchptrloc ← $helploc;                                         06635
      exflagptrloc ← 1; % Flag for execution rather than goto. %
                                                                         06636
      xgoto(0, parsing, $subsysptrloc, $dispatchptrloc,
      $exflagptrloc);                                                    06637
   % stuff a cmdelete character into the input buffer %                  06638
      !sti(18M, CD);                                                     06639
   % resume execution %                                                 06640
      RETURN;                                                            06641
   END.
                                                                         06642
(dfnsubsys) PROCEDURE( % defines a subsystem by adding/replacing a
subsystem to/from the passed subsystem dispatch stack %                  05550
   % FORMAL ARGUMENTS %                                                 05551
      dptptr          % ptr to subsystem dispatch record (this is the
      symbolic name which appears on the SUBSYSTEM statement of the
      CML definition of the subsystem %                                  05552
      controlbits,  % TNLS, DNLS, level 1 options %                     05553
      subrule);     %ptr to subsystem rule (either nlssubs o°
      allsubs)%                                                          05554
   % RETURNS %                                                          05555
      % none %                                                          05556
   % ABNORMAL RETURNS %                                                 05557
      % SIGNALS interperr if stack overflows %                          05558
   LOCAL % VARIABLES %                                                   05559
      frameptr,       % ptr to current frame in the allsubs %           05560
      instptr,        % ptr to CML instruction record %                 05561
      pdptx,          %ptr to size of subrule %                        05562
      i;     % index counter %                                          05563
   REF % VARIABLES %                                                     05564
      dptptr, frameptr, instptr, pdptx, subrule;                         05565
   % ---------------------- %                                           05566
   % check to make sure that we've been given a valid pointer %         05567
      IF dptptr.dptvalid # $dptvldationcode THEN                         05568
```

```
      err( $"Invalid Subsystem Identifier");                          05569
%check to be sure we are passed a valid subrule and set up pdptx%
                                                                      05570
   CASE &subrule OF                                                   05571
      = $nlssubs: &pdptx ← $sdptx;                                    05572
      = $allsubs: &pdptx ← $sdptxa;                                   05573
      ENDCASE err($"Invalid subsystem rule passed to dfnsubsys");
                                                                      05574
% initialize the subsystem dispatch stack if it has not already
been done %                                                           05575
   IF pdptx < 4 THEN                                                  05576
      BEGIN                                                           05577
      &instptr ← &subrule; % build $xecute instruction %              05578
      FOR &frameptr ← &instptr UP UNTIL > &instptr+3 DO               05579
         % initialize the first 4 words to zeroes %                   05580
          frameptr ← 0;                                               05581
      % build an execute instruction in the first position %          05582
         instptr.opcode ← $xecute;                                    05583
      % build a STORE insturction in the next position %              05584
         &instptr ← &instptr + 2;                                     05585
         instptr.opcode ← $store;                                     05586
         instptr.addr ← $grammar;                                     05587
      % set pdptx to point to next loc in dispatch stack %            05588
         pdptx ← 4;                                                   05589
      END;                                                            05590
% search through the dispatch stack and set frameptr to point to
the frame to be replaced.  If none is found, look for a null
frame (from a previous deleted subsystem) %                           05591
   FOR &frameptr ← &subrule+4 UP $framesize UNTIL >=
   &subrule+pdptx DO                                                  05592
      CASE frameptr.opcode OF                                         05593
         = $keyop:       % try to replace existing subsystem %
                                                                      05594
            IF */frameptr.addr/* = */dptptr.dptname/* THEN            05595
               BEGIN                                                  05596
               &instptr ← &frameptr;                                  05597
               instptr.addr ← dptptr.dptname;                         05598
               instptr.ctrl ← controlbits;                            05599
               &instptr ← &instptr + 2;                               05600
               instptr.addr ← &dptptr;                                05601
               RETURN;                                                05602
               END;                                                   05603
         ENDCASE;                                                     05604
% search through the dispatch stack and set frameptr to point to
a null frame (from a previous deleted subsystem) %                    05605
   FOR &frameptr ← &subrule+4 UP $framesize UNTIL >=
   &subrule+pdptx DO                                                  05606
      CASE frameptr.opcode OF                                         05607
         = 0:    % try to replace null frame %                        05608
            GOTO buildit;                                             05609
         ENDCASE;                                                     05610
% allocate a new frame at the end of the stack %                      05611
   &frameptr ← &subrule+pdptx;                                        05612
   IF pdptx >= $sdptsize THEN                                         05613
      SIGNAL(interperr, $"Subsystem dispatch stack overflowed");
                                                                      05614
```

```
      pdptx ← pdptx + $framesize;                                    05615
   (buildit):                                                        05616
   % initialize the frame to 0 %                                     05617
      FOR &instptr ← &frameptr UP UNTIL >= &frameptr+$framesize DO
                                                                     05618
         instptr ← 0;                                                05619
   % build a keyword recognition instruction - set the successor to
   point to the next instruction (to be subsequently built) and set
   the alternative path to zero %                                    05620
      &instptr ← &frameptr;                                          05621
      instptr.nsuccessor ← &frameptr+2;                              05622
      instptr.opcode ← $keyop;                                       05623
      instptr.addr ← dptptr.dptname;                                 05624
      instptr.ctrl ← controlbits;                                    05625
   % build an $enter value instruction - the alternative path is
   null as this is the last inst. in the stack.  The successor
   points to the STORE instruction in the top of the stack %         05626
      &instptr ← &frameptr+2;                                        05627
      instptr.nsuccessor ← &subrule + 2;                             05628
      instptr.opcode ← $enter;                                       05629
      instptr.addr ← &dptptr;                                        05630
   % link the new entry into the existing stack.  If this is the
   first entry on the stack, then the EXECUTE instruction at the top
   of the stack must point to here, otherwise this frame is linked
   as an alternative to the previous frame %                         05631
      IF &frameptr = &subrule + $framesize                           05632
         THEN % first entry %                                        05633
            BEGIN                                                    05634
            frameptr.alternative ← subrule.addr;                     05635
            subrule.addr ← &frameptr                                 05636
            END                                                      05637
         ELSE                                                        05638
            BEGIN % subsequent entry %                               05639
            &instptr ← &frameptr - $framesize;                       05640
            frameptr.alternative ← instptr.alternative;              05641
            instptr.alternative ← &frameptr;                         05642
            END;                                                     05643
   RETURN;                                                           05644
   END.
                                                                     05645
(delsubsys) PROCEDURE( % deletes a subsystem by nulling its entry in
the subsystem rule subrule %                                         04980
   % FORMAL ARGUMENTS %                                              04981
      dptptr,        % ptr to subsystem dispatch record (this is the
      symbolic name which appears on the SUBSYSTEM statement of the
      CML definition of the subsystem %                              04982
      subrule);      %ptr to a subsystem rule (nlssubs or allsubs)%
                                                                     04983
   % RETURNS %                                                       04984
      % TRUE if successful, FALSE subsystem not found in the nlssubs
      stack %                                                        04985
   % ABNORMAL RETURNS %                                             04986
      % calls err if actual argument is not valid %                 04987
   LOCAL % VARIABLES %                                               04988
      frameptr,      % ptr to current frame in the nlssubs %         04989
      instptr,       % ptr to CML instruction record %               04990
```

```
      savalt,        %used to save alternative of deleted entry% 04991
      i;     % index counter %                                   04992
   REF % VARIABLES %                                            04993
      dptptr, frameptr, subrule, instptr;                       04994
   % -------------------- %                                     04995
   % check to make sure that we've been given a valid pointer % 04996
      IF dptptr.dptvalid # $dptvldationcode THEN                04997
         err( $"Invalid =subsystem Identifier");                 04998
   % search through the dispatch stack and set frameptr to point to
   the frame to be deleted. %                                   04999
      &frameptr ← &subrule;                                     05000
      IF frameptr.opcode NOT= $xecute THEN RETURN( FALSE );     05001
      &frameptr ← frameptr.addr;                                05002
      IF *[frameptr.addr]* = *[dptptr.dptname]* THEN            05003
         BEGIN                                                  05004
         subrule.addr ← frameptr.alternative;                   05005
         % zap the four word entry to zeoes %                   05006
            FOR &instptr ← &frameptr UP UNTIL >=
            &frameptr+$framesize DO                             05007
               instptr ← 0;                                     05008
         RETURN( TRUE );                                        05009
         END;                                                   05010
      WHILE &frameptr DO                                        05011
         IF frameptr.alternative AND [frameptr.alternative].opcode =
         $keyop THEN                                            05012
            IF *[[frameptr.alternative].addr]* = *[dptptr.dptname]*
            THEN                                                05013
               BEGIN                                            05014
               savalt ← [frameptr.alternative].alternative;     05015
               % zap the four word entry to zeoes %             05016
                  FOR &instptr ← frameptr.alternative UP UNTIL >=
                  frameptr.alternative+$framesize DO            05017
                     instptr ← 0;                               05018
               frameptr.alternative ←savalt;                    05019
               RETURN( TRUE );                                  05020
               END                                              05021
            ELSE &frameptr ← frameptr.alternative               05022
         ELSE RETURN( FALSE );                                  05023
   RETURN (FALSE);                                              05024
   END.
                                                                05025
(supervisor) PROC; % main control routine %                     05689
   % supervisor is the principal control routine for NLS.  It
   maintains a stack of subsystem executions and passes control to
   the command interpreter %                                    05690
   % FORMAL ARGUMENTS %                                         05691
      % none %                                                  05692
   % RETURN VALUES %                                           05693
      % none -- this routine maintains control until we quit NLS
      altogether, at which time it simply returns to its caller %
                                                                05694
   LOCAL % VARIABLES %                                          05695
      sdispptr,      % ptr to subsystem dispatch record %       05696
      count,         % iteration count for subsystem %          05697
      entry,         % ptr to current sbstack entry %           05698
      ptr,           % starting point for cmdinterpretter %     05699
```

```
    instptr,        % ptr to dummy interpretter instructions %  05700
    insts[4];       % dummy grammar to connect supervisor to
    subsystem %                                                 05701
LOCAL STRING                                                    05702
    locstr[150],            %used for file names%               05703
    sysnamestr[50];         % subhelp string for subsystems %   05704
REF % VARIABLES %                                               05705
    entry,                                                      05706
    instptr,                                                    05707
    sdispptr;                                                   05708
LOCAL STRING errstr[50];                                        05709
%-------------%                                                 05710
% initialize the NLS subsystems if they have not already been
initialized .  If we get an error here, just abort by returning %
                                                                05711
    ON SIGNAL                                                   05712
        ELSE RETURN;                                            05713
    IF sdptx <= 0 THEN initsubsystems();                        05714
% trap all signals and put out error disgnostics %              05715
    ON SIGNAL                                                   05716
        = gaderr:  % address expression error %                 05717
            BEGIN                                               05718
            *errstr* ← *[MESSAGE]*, " ??";                      05719
            dismes(2, $errstr);                                 05720
            GOTO errline;                                       05721
            END;                                                05722
    ELSE                                                        05723
        BEGIN                                                   05724
        IF &sysmsg AND sysmsg.L THEN                            05725
            IF sysmsg.LH < sysmsg.RH                            05726
                THEN  dismes(2, $"system screwup, invalid value for
                sysmsg")                                        05727
                ELSE  dismes(2,&sysmsg);                        05728
        (errline):                                              05729
        &sysmsg ← 0;                                            05730
        IF auxinflag THEN auxinterminate();                     05731
        GOTO mainline;                                          05732
        END;                                                    05733
% initialize the system message pointer to null %              05734
    &sysmsg ← 0;                                                05735
% initialize some interpreter control stuff %                  05736
    sbstkx ← 0;                                                 05737
% construct an initial sbstack entry:  Use the first entry in the
NLSSUB stack of subsystems as the default subsystem %           05738
    &instptr ← $nlssub+6; % ptr to first ENTER instruction %    05739
    % check to make sure we've really got an $enter instruction %
                                                                05740
        IF instptr.opcode # $enter THEN RETURN; % abort %       05741
    &sdispptr ← instptr.addr;                                   05742
    &entry ← $sbstack;                                          05743
    entry.sbptr ← &sdispptr;                                    05744
    entry.sbcount ← -1;                                         05745
    entry.sbmode ← sbstart;                                     05746
    entry.sbnptr ← sdispptr.dptname;                            05747
    sbstkx ← $sbentsize;                                        05748
% initialize the insts to connect a subsystem grammar with the
```

```
supervisor's grammar %                                                  05749
   CASE &sdispptr ← getsdptr($usysl,$allsubs) OF                        06307
      = 0; %not an nls subsystem must be a user subsystem%              06308
         BEGIN                                                          06309
            ON SIGNAL                                                   06310
            ELSE                                                        06311
               BEGIN                                                    06312
               ON SIGNAL ELSE;                                          06313
               *errstr* ← *usysl* , "  not available: using nls
               default supervisor";                                     06314
               dismes(2,$errstr);                                       06315
               &sdispptr ← $subsupervisor;                              06316
               EXIT CASE;                                               06317
               END;                                                     06318
            getuprog($usysl);                                           06319
            IF &sdispptr ← getsdptr($usysl,$nlssubs) THEN               06320
               delsubsys(sdispptr,$nlssubs)%gpget has defined
               supervisor as a subsystem , we can't allow user to
               go to his supervisor subsystem%                          06321
            ELSE err($errstr); %the specified user program was
            not a subsystem%                                            06322
         END;                                                           06323
   ENDCASE;                                                             06324
   FOR &instptr ← $insts UP UNTIL = $insts+4 DO instptr ← 0;            05765
   &instptr ← $insts;                                                   05766
   instptr.opcode ← $xecute;                                            05767
   &instptr ← instptr.alternative ← $insts + 2;                         05768
   instptr.opcode ← $xecute;                                            05769
   instptr.addr ← sdispptr.dptrun;                                      05770
   &instptr ← $insts;                                                   05771
dpset(dspno, endfil, endfil, endfil);                                   05772
cmdmode ← 0;                                                            05773
IF nlmode                                                               05774
   THEN cmdmode.dnlscmd ← TRUE                                          05775
   ELSE cmdmode.tnlscmd ← TRUE;                                         05776
IF recogmode = mexpert                                                  05777
   THEN cmdmode.llcmd ← TRUE;                                           05778
(mainline): % main process control loop %                               05779
LOOP                                                                    05780
   BEGIN                                                                05781
   % process the top entry in the subsystem stack; If the stack
   is empty then we exit the process loop %                             05782
      IF sbstkx <= 0 THEN EXIT LOOP;                                    05783
      &entry ← $sbstack + sbstkx - $sbentsize;                          05784
      &sdispptr ← entry.sbptr;                                          05785
   % process by the mode of the entry %                                 05786
      CASE entry.sbmode OF                                              05787
         = sbstart:      % initialization %                             05788
            BEGIN                                                       05789
            ptr ← sdispptr.dptinit; % ptr to initialization rule
            %                                                           05790
            entry.sbmode ← sbrun;                                       05791
            % set up some subsystem prompting stuff %                   05792
               sethrld( entry.sbnptr );                                 05793
            % reset command repeat string %                            05794
               *keyrptstr* ← NULL;                                      05795
```

```
            END;                                                    05796
         = sbrun:        % executing is a subsystem %               05797
            BEGIN                                                   05798
            IF (entry.sbcount ← entry.sbcount-1 ) = 0               05799
               THEN entry.sbmode ←sbfinish;                         05800
            instptr.addr ← sdispptr.dptrun; % ptr to subsystem
            rule %                                                  05801
            ptr ← $insts;                                           05802
            cuefig ← FALSE;                                         05803
            fbcti( clearcfl );                                      05804
            % setup for typewriter %                               05805
               IF nlmode = typewriter THEN                          05806
                  BEGIN                                             05807
                  FOR count ← 1 UP UNTIL > fedind DO typecn(SP);
                                                                    05808
                  typeas( $hrldstr ); % type out the hearald %
                                                                    05809
                  END;                                              05810
            END;                                                    05811
         = sbfinish:      % termination of the subsystem %          05812
            BEGIN                                                   05813
            ptr ← sdispptr.dptfinish; % ptr to termination rule %
                                                                    05814
            entry.sbmode ← sbpop;                                   05815
            END;                                                    05816
         = sbpop:         % pop subsystem %                         05817
            BEGIN                                                   05818
            sbstkx ← sbstkx - $sbentsize;                           05819
            % set up some subsystem prompting stuff %              05820
               IF sbstkx > 0 THEN                                   05821
                  BEGIN                                             05822
                  &entry ← $sbstack + sbstkx - $sbentsize;          05823
                  sethrld( entry.sbnptr );                          05824
                  END;                                              05825
            % reset command repeat string %                        05826
               *keyrptstr* ← NULL;                                  05827
            REPEAT LOOP;                                            05828
            END;                                                    05829
         = sbrentry:      % reset after leaving subsystem %         05830
            BEGIN                                                   05831
            ptr ← sdispptr.dptrentry;                               05832
            % reset previous mode %                                05833
               entry.sbmode ← entry.sbpmode;                        05834
            END;                                                    05835
         ENDCASE                                                    05836
            BEGIN                                                   05837
            dismes( 2,$"interpreter mode screwup, aborted");        05838
            RETURN;                                                 05839
            END;                                                    05840
      % invoke the interpreter to evaluate the rule if there is one
      %                                                             05841
      IF ptr THEN                                                   05842
         BEGIN                                                      05843
         % set up for new command %                                05844
            pathx ← evalx ← ptrx ← 0;                               05845
            keyinpstr.L ← 0;                                        05846
```

```
            keysaveflag ← basestateflag ← TRUE;                          05847
            IF NOT &tda THEN &tda ← lda();                               05848
            IF nlmode = typewriter THEN                                  05849
               BEGIN                                                     05850
               cspupdate ← &tda;                                         05851
               curmkr ← tda.dacsp;                                       05852
               curmkr[1] ← tda.daccnt;                                   05853
               END                                                       05854
            ELSE cspupdate ← 0;                                          05855
            cspvs ← tda.davspec;  cspvs[1] ← tda.davspc2;                05856
            cspusqcod ← tda.dausqcod;                                    05857
            cspcacode ← tda.dacacode;                                    05858
         % invoke the interpreter %                                      05859
            cmdinterp( $pathstk+3, $ptr );                               05860
         % reset after successful command completion %                   05861
            (cmdend):                                                    05862
            cmdfinish();                                                 05863
         END;                                                            05864
      END;                                                               05865
   % if we fall out of the main loop, then the subsystem stack must
   be empty and we are exiting the system %                              05866
      RETURN;                                                            05867
   END.
                                                                         05868

(cmdfinish) PROCEDURE;                                                    0535
   REF tda;                                                              03407
   LOCAL srr, frr, stid, cc;  REF frr, srr;                               0536
   LOCAL STRING locstr[200];                                              0537
   IF cspupdate THEN                                                      0538
      BEGIN                                                               0539
      &tda ← cspupdate;                                                   0540
      stid ← tda.dacsp;                                                   0541
      cc ← tda.daccnt;                                                    0542
      %update statement return ring%                                      0543
         &frr ← tda.dalink;  %get address of file return ring%            0544
         IF NOT frr.frhexis THEN                                          0545
            err($"Illegal file return ring detected in cmdfinish");
                                                                          0546
         %get frr  entry address%                                         0547
            &frr ← &frr + frrhlen + (frrelen*frr.frhtop);                 0548
         IF frr.frexis AND NOT tda.daempty AND tda.dacsp NOT= endfil
         THEN %update srr%                                                0549
            BEGIN                                                         0550
            %get address of statement return ring%                        0551
               &srr ← frr.frsrring;                                       0552
            %update old position and viewspecs on ring%                   0553
               storesrring(&srr, 0, tda.dacsp, tda.daccnt,
               tda.davspec, tda.davspc2);                                 0554
                  %user may have changed viewspecs%                       0555
            END;                                                          0556
      IF curmkr.stfile NOT= tda.dacsp.stfile THEN  %changing files,
      push file return ring%                                              0557
         BEGIN                                                            0558
         %get name of new file%                                           0559
            *locstr* ← NULL;                                              0560
            filnam(curmkr.stfile, $locstr);                               0561
```

```
        %push new file name on ring%                                    0562
           pushfrring(tda.dalink, $locstr, curmkr.stfile);              0563
           readfrring(tda.dalink, 0 : &srr);                            0564
           IF usesrr THEN  %jump file return -- copy usesrr to new
           srr%                                                         0565
              BEGIN                                                     0566
              copysrring(usesrr, &srr);                                 0567
              END;                                                      0569
        %put out "modified" message if necessary%                       0570
           IF [flntadr(curmkr.stfile)].fllock THEN                      0571
              lockmes(curmkr.stfile)                                    0572
           ELSE dismes(2, $locstr);  %show user new file name%          0573
        %close files no longer used in display areas%                   0574
           tda.dacsp ← curmkr;                                          0575
           tda.daccnt ←                                                 0576
              IF nlmode = fulldisplay THEN 1 ELSE curmkr[1];            06334
           tda.daempty ← FALSE;                                         0577
           freflnt(); %close files no longer needed%                    0578
        END;                                                            0579
     tda.dacsp ← curmkr;                                                0580
     tda.daccnt ←                                                       06335
        IF nlmode = fulldisplay THEN 1 ELSE curmkr[1];                  06336
     tda.daempty ← FALSE;                                               0582
     %update viewspecs%                                                 0583
        tda.dapvs ← tda.davspec := cspvs;                               0584
        tda.dapvs2 ← tda.davspc2 := cspvs[1];                           0585
        tda.dacacode ← cspcacode;                                       0586
        tda.dausqcod ← cspusqcod;                                       0587
     %push new position and viewspecs onto statement return ring%
                                                                        0588
        IF NOT usesrr := 0 AND (stid NOT= curmkr OR cc NOT=
        curmkr[1]) THEN                                                 0589
           pushsrring(&srr, tda.dacsp, tda.daccnt, tda.davspec,
           tda.davspc2);                                                0590
                                                                        0591
     END;                                                               0592
  IF nlmode = fulldisplay THEN                                          0593
     BEGIN                                                              0594
     % recreate the display %                                          0595
        recred();                                                       0596
        cdtype ← dspno; % shut off recred until dpset is called
        again %                                                         0597
     IF lplitreset THEN                                                 07136
        BEGIN                                                           07137
        litline ← lplitline;                                            07138
        litreset ← FALSE;                                               07139
        litapflag ← TRUE;                                               07140
        rstlit();                                                       07141
        lplitreset ← FALSE;                                             07142
        END;                                                            07143
     END;                                                               0598
  ckrrings(); %to help find the bug that is smashing the file and
  statement return rings%                                               06530
  RETURN;                                                               0599
  END.                                                                  0600
                                                                        0601
```

```
(ckrrings) PROC; %check consistency of return rings%                    06531
   LOCAL da, end, frr, srr, frrend;                                     06535
   REF da, frr, srr;                                                    06536
   end ← (&da ← $dpyarea) + dal*dacnt;                                  06537
   DO IF da.daexis THEN                                                 06538
      BEGIN                                                             06540
      IF NOT (&frr ← da.dalink) THEN                                    06542
         werr(S"return ring error: dalink empty");                      06543
      IF NOT frr.frhexis THEN                                           06544
         werr(S"return ring error: FRR empty");                         06551
      frrend ← &frr + frr.frhlast*frrelen + frrhlen;                    06556
      FOR &frr ← &frr + frrhlen UP frrelen UNTIL >=frrend DO            06545
         IF frr.frexis THEN                                             06546
            BEGIN                                                       06553
            IF NOT (&srr ← frr.frsrring) THEN                           06547
               werr($"return ring error: FRSRING field empty");         06554
            IF NOT srr.srhexis THEN                                     06548
               werr($"return ring error: SRR empty");                   06555
            END;                                                        06552
      END                                                               06541
   UNTIL (&da ← &da + dal) >= end;                                      06539
   RETURN;                                                              06532
   END.                                                                 06533
                                                                        06534
(sethrld) PROC( % sets up command hearald based upon subsystem name
%                                                                       0602
   % FORMAL ARGUMENTS %                                                 0603
      strptr);       % ptr to hearald name string %                    0604
   REF strptr;                                                          0605
   %-------------%                                                      0606
   % save subsystem name away in global string ssysname %              0607
      *ssysname* ← *strptr*;                                            0608
   CASE hrldmode OF                                                     0611
      = onechar:     % single char hearald %                           0612
         *hrldstr* ← "* ";                                              0613
      = multchar:    % multiple char hearald %                         0614
         *hrldstr* ← *strptr* [1 TO MIN(hrldsize, strptr.L,
         hrldstr.M-1)], SP;                                             0615
      ENDCASE;                                                          0616
   IF nlmode = fulldisplay                                              0609
      THEN                                                              0617
         BEGIN       % display subsystem name %                         0618
         dsubsys( &strptr );                                            0619
         END;                                                           0620
   RETURN;                                                              0621
   END.
                                                                        0622
% COMMAND INTERPRETER %                                                0623
   % MAIN CONTROL ROUTINES %                                           0624
      (cmdinterp) PROCEDURE(                                            02982
         % FORMAL ARGUMENTS %                                           02983
            resultptr,          % pointer to the result record %       02984
            argptr );  %pointer to a function state record containing a
            pointer to a node in the grammar %                          02985
         % NORMAL RETURNS %                                            02986
            % 1) pointer to a function sta@e record of the result,
```

```
   This pointer is set to 0 if the parse fails. %                    02987
% ABNORMAL RETURNS %                                                 02988
   % abnormal returns are accomplished via SIGNALS, generated
   for the following conditions %                                    02989
      % 1) interperr -- an interpreter failure is detected %
                                                                     02990
      % 2) cmddelete -- the command is aborted due to a
      command delete char. %                                         02991
LOCAL % VARIABLES %                                                 02992
   tempptr,    % temporary pointer, working value %                 02993
   instptr,    % ptr to interpretive text instruction %             02994
   curptr,     % ptr to current path stack entry %                  02995
   nextptr,    % ptr to next path stack entry %                     02996
   firstptr,   % ptr to first path stack entry %                    02997
   stateptr,   % ptr to function state record %                     02998
   pathptr,    % ptr to staT OF SELECTION PATH /                   02999
               % = 0 if selection just completed %                  03000
   function;   % address of processing function %                  03001
LOCAL STRING locstr[100];                                           03002
REF % VARIABLES %                                                   03003
   pathptr,                                                         03004
   tempptr,                                                         03005
   instptr,                                                         03006
   function,                                                        03007
   curptr,                                                          03008
   nextptr,                                                         03009
   firstptr,                                                        03010
   stateptr,                                                        03011
   resultptr,                                                       03012
   argptr;                                                          03013
%-------------------------%                                        03014
% trap any state changing signals %                                 03015
   ON SIGNAL                                                        03016
      = popstate:                                                   03017
         BEGIN                                                      03018
         % curptr points to a selection function frame.  We
         want to back out of this frame, to the beginning of
         the previous frame if there is one %                       03019
            WHILE &curptr >= &pathptr DO                            03020
               % back out of current frame %                        03021
               BEGIN                                                03022
               % check for backup in/into a selection function
               %                                                    03023
                  IF testselect( &curptr ) THEN                     03024
                     BEGIN                                          03025
                     &nextptr ← &curptr + $totalrecsize;            03026
                     &instptr ← curptr.begnodeptr;                  03027
                     &pathptr ← curptr.markptr;                     03028
                     pathx ← &nextptr - $pathstk;                   03029
                     GOTO parseit;                                  03030
                     END;                                           03031
               % $call processing function in "cleanup" mode %
                                                                    03032
                  IF (&function ← curptr.pfunction) # 0  THEN
                                                                    03033
                     function( &curptr+$pathrecsize, cleanup
```

```
            );                                                  03034
         % back curptr down to previous frame %                 03035
            &curptr ← &curptr - $totalrecsize;                  03036
         END;                                                   03037
      % curptr now points to the frame preceeding the top
      one (if one exists).  set up local interpreter
      variables and the global state variables and resume
      the parse at the top position of the current path in
      the grammar %                                             06306
      LOOP                                                      03039
         BEGIN                                                  03040
         % reset the prompting flag %                           03041
            cueflg ← FALSE;                                     03042
         % collapse the path stack down to the last
         selection function %                                   03043
            &nextptr ← &curptr ← &curptr + $totalrecsize;
                                                                03044
            pathx ← &nextptr - $pathstk;                        03045
            ptrx ← curptr.ptrxsav;                              03046
            evalx ← curptr.evalxsav;                            03047
         % let the signal propagate if we have reached the
         bottom of the path stack %                             03048
            IF &curptr <= &firstptr THEN EXIT LOOP;             03049
         % back the path stack up to the beginning of the
         previous selection %                                   03050
            &curptr ← ^curptr - $totalrecsize;                  03051
            &pathptr ← curptr.markptr;                          03052
         % curptr points to a selection function frame.  We
         want to back out of this frame, to the beginning
         of the previous frame if there is one %                03053
            WHILE &curptr >= &pathptr DO                        03054
               % back out of current frame %                    03055
               BEGIN                                            03056
               % check for backup in/into a selection
               function %                                       03057
                  IF testselect( &curptr ) THEN                 03058
                     BEGIN                                      03059
                     &nextptr ← &curptr + $totalrecsize;
                                                                03060
                     &instptr ← curptr.begnodeptr;              03061
                     &pathptr ← curptr.markptr;                 03062
                     pathx ← &nextptr - $pathstk;               03063
                     GOTO parseit;                              03064
                     END;                                       03065
               % $call processing function in "cleanup"
               mode %                                           03066
                  IF (&function ← curptr.pfunction) # 0
                                                                03067
                     THEN function( &curptr+$pathrecsize,
                     cleanup );                                 03068
               % back curptr down to previous frame %           03069
                  &curptr ← &curptr - $totalrecsize;            03070
               END;                                             03071
         %check if ignoring level adjust or viewspecs %
                                                                03072
            IF nolevadj AND [pathptr.curnodeptr].opcode =
```

```
            $levadj                                                 03073
               THEN REPEAT LOOP;                                    03074
            IF novspec AND [pathptr.curnodeptr].opcode =
            $vwspecs                                                03075
               THEN REPEAT LOOP;                                    03076
         % set up to resume execution %                             03077
            % collapse the path stack %                             03078
               &nextptr ← &curptr ← &curptr +
               $totalrecsize;                                       03079
               pathx ← &nextptr - $pathstk;                         03080
            &instptr ← curptr.begnodeptr;                           03081
            &pathptr ← 0;                                           03082
            ptrx ← curptr.ptrxsav;                                  03083
            evalx ← curptr.evalxsav;                                03084
            curptr.pmode ← parsing;                                 03085
            fbctl( fbpop );                                         03086
            GOTO parseit;                                           03087
         END;                                                       03088
      END;                                                          03089
= cutpathstk:                                                       06232
      BEGIN % cut stack back to frame indicated by cutstop
      %                                                             06233
      IF cutstop > 0 THEN                                           06305
         WHILE &curptr > cutstop DO                                 06235
            % back out of current frame %                           06236
            BEGIN                                                   06237
            % call processing function in "cleanup" mode %
                                                                    06247
               IF (&function ← curptr.pfunction) # 0  THEN
                                                                    06248
                  function( &curptr+$pathrecsize, cleanup
                  );                                                06249
            % back curptr down to previous frame %                  06250
               &curptr ← &curptr - $totalrecsize;                   06251
            END;                                                    06252
      % curptr now points to the frame specified by
      cutstop.  set up local interpreter variables and the
      global state variables and resume the parse at the
      top position of the current path in the grammar %
                                                                    06253
      LOOP                                                          06254
         BEGIN                                                      06255
         % reset the prompting flag %                               06256
            cueflg ← FALSE;                                         06257
         % collapse the path stack down to the last
         selection function %                                       06258
            &nextptr ← &curptr ← &curptr + $totalrecsize;
                                                                    06259
            pathx ← &nextptr - $pathstk;                            06260
            ptrx ← curptr.ptrxsav;                                  06261
            evalx ← curptr.evalxsav;                                06262
         % let the signal propagate if we have reached the
         bottom of the path stack %                                 06263
            IF &curptr <= &firstptr THEN EXIT LOOP;                 06264
         % back the path stack up to the beginning of the
         previous frame %                                           06265
```

```
                &curptr ← &curptr - $totalrecsize;                    06266
                &pathptr ← curptr.markptr;                            06267
             %check if ignoring level adjust or viewspecs %
                                                                      06267
                IF nolevadj AND (pathptr.curnodeptr).opcode =
                $levadj                                               06268
                   THEN REPEAT LOOP;                                  06269
                IF novspec AND (pathptr.curnodeptr).opcode =
                $vwspecs                                              06270
                   THEN REPEAT LOOP;                                  06271
             % set up to resume execution %                           06272
                % collapse the path stack %                           06273
                   &nextptr ← &curptr ← &curptr +
                   $totalrecsize;                                     06274
                   pathx ← &nextptr - $pathstk;                       06275
                &instptr ← curptr.begnodeptr;                         06276
                &pathptr ← 0;                                         06277
                ptrx ← curptr.ptrxsav;                                06278
                evalx ← curptr.evalxsav;                              06279
                curptr.pmode ← parsing;                               06280
                fbctl( fbpop );                                       06281
                GOTO parseit;                                         06282
             END;                                                     06283
          END;                                                        06284
       ELSE % all other signals must be error/command delete
       signals %                                                      03090
          BEGIN                                                       03091
          (abort):                                                    03092
          WHILE &curptr >= &firstptr DO                               03093
             BEGIN                                                    03094
             % $call processing function in "cleanup" mode %
                                                                      03095
                IF (&function ← curptr.pfunction) # 0                 03096
                   THEN function( &curptr+$pathrecsize, cleanup
                   );                                                 03097
             % cleanup to previous path stack entry and
             collapse the stack %                                     03098
                &curptr ← &curptr - $totalrecsize;                    03099
             END;                                                     03100
          % reset the prompting flag %                               03101
             cueflg ← FALSE;                                          03102
          % reset completion code %                                  03103
             complcode ← 1;                                           03104
          END;                                                        03105
    % initialize interpreter variables %                             03106
       &instptr ← argptr;                                             03107
       &curptr ← &nextptr ← &firstptr ← $pathstk + pathx;             03108
       &pathptr ← 0;                                                  03109
    % initialize local variables %                                   03110
       curptr.pmode ← parsing;                                        03111
    (parseit):                                                        03112
    % continue parsing as long as we are doingit %                   03113
       WHILE &curptr DO                                               03114
          BEGIN                                                       03115
          CASE curptr.pmode OF % process according to type of
          parse %                                                     03116
```

```
= parsing:              % normal parsing mode %      03117
  BEGIN                                             03118
  % set up a new path stack entry %                 03119
     &curptr ← &nextptr;                            03120
     curptr.begnodeptr ← &instptr;                  03121
     CASE &nextptr ← &nextptr + $totalrecsize OF
                                                    03122
        >= $pathstk + $pssize:                      03123
           SIGNAL (interperr, $"Path Stack
           Overflowed");                            03124
        > $pathstk + pathx:                         03125
           BEGIN                                    06643
           % no keyword recognition yet %           06646
              kwrstate ← 0;                         06644
           pathx ← &nextptr - $pathstk;             03126
           END;                                     06645
     ENDCASE;                                       03127
     % initialize frame values which are not altered
     after backup %                                 03128
        curptr.ptrxsav ← ptrx;                      03129
        curptr.evalxsav ← evalx;                    03130
     (resume):  % resume here after backup %        03131
     % set pathptr to curptr if not already set
     do not update pathptr if next inst is a STORE %
                                                    03132
        IF NOT &pathptr AND instptr.opcode # $store
                                                    03133
           THEN &pathptr ← &curptr;                 03134
     % bump frame counter (10 bits resolution) %
                                                    03135
        framecounter ← (framecounter+1) .A 1777B;
                                                    03136
     % initialize the path record %                 03137
        curptr.curnodeptr ← &instptr;               03138
        curptr.pmode ← parsing;                     03139
        curptr.evalmod ← $unknown;                  03140
        curptr.fcounter ← framecounter;             03141
        curptr.pfunction ← 0;                       03142
        curptr.markptr ←                            03143
           IF &pathptr THEN &pathptr                03144
           ELSE (&curptr - $totalrecsize).markptr;
                                                    03145
  % set function state ptr %                        03146
     &stateptr ← &curptr + $pathrecsize;            03147
  % evaluate the upcoming node %                    03148
     evaler( &curptr );                             03149
  END;                                              03150
= cleanup:              % termination of a command %
                                                    03151
  BEGIN                                             03152
  % save current keyword save string for possible
  upcoming repeat command function %                03153
     *keyrptstr* ← *keyinpstr*;                     03154
  % backup in the command as far as necessary
  (depending on the type of temination %            03155
     &tempptr ← IF complcode = 1                    03156
```

```
                THEN &firstptr                                  03157
                ELSE setbackup(&firstptr, &curptr);             03158
             WHILE &curptr >= &tempptr DO                       03159
                BEGIN                                           03160
                % $call the execution function in cleanup
                mode if one exists %                            03161
                   IF (&function ← curptr.pfunction ) # 0
                                                                03162
                      THEN function ( &curptr+$pathrecsize,
                      cleanup);                                 03163
                &curptr ← &curptr - $totalrecsize;              03164
                END;                                            03165
          % terminate processing, prepare for rpt or insert
          if required %                                         03166
             CASE complcode OF                                  03167
                = 1:          % normal CA %                     03168
                   RETURN;                                      03169
                = 2:          % insert statement mode %         03170
                   BEGIN                                        03171
                   &curptr ← &tempptr;                          03172
                   &instptr ← $zinsstatement;                   03173
                   curptr.begnodeptr ← &instptr;                03174
                   END;                                         03175
                = 3:          % repeat command mode %           03176
                   BEGIN                                        03177
                   &curptr ← &tempptr;                          03178
                   &instptr ← curptr.begnodeptr;                03179
                   IF nlmode = fulldisplay THEN cflasp();
                                                                03180
                   END;                                         03181
                ENDCASE err(notyet);                            03182
          % prepare to resume execution %                       03183
             ptrx ← curptr.ptrxsav;                             03184
             evalx ← curptr.evalxsav;                           03185
             &nextptr ← &curptr + $totalrecsize;                03186
             &pathptr ← 0;                                      03187
             cueflg ← FALSE;                                    03188
          cmdfinish();                                          03189
          IF NOT &tda THEN &tda ← lda();                        03320
          cspupdate ← IF nlmode = typewriter THEN &tda ELSE
          0;                                                    03321
          % following two intructions nopped by kev 6/4/74 %
                                                                03327
             % curmkr ← tda.dacsp; %                            03322
             % curmkr[1] ← tda.daccnt; %                        03323
          cspvs ← tda.davspec;  cspvs[1] ← tda.davspc2;         03324
          cspusqcod ← tda.dausqcod;                             03325
          cspcacode ← tda.dacacode;                             03326
          GOTO resume;                                          03190
          END;                                                  03191
       = pnext:                                                 03192
          BEGIN                                                 03193
          basestateflag ← FALSE; % no longer in base command
          state %                                               03194
          &instptr ← curptr.curnodeptr;                         03195
          % reset pathptr if last instruction was a
```

```
                selection function %                                    03196
                   IF instptr.opcode IN (Skeyop, $levadj)               03197
                      THEN                                              03198
                         BEGIN                                          03199
                         &patnptr ← 0;                                  03200
                         lastsel ← instptr.opcode;                      03201
                         END;                                           03202
                % stop collecting keyword input strings (for
                possible command repeat) if we just finished a
                non-keyword recognizer %                                03203
                   IF instptr.opcode IN (Sconfirm, $call)               03204
                      THEN keysaveflag ← FALSE;                         03205
                CASE &instptr ← instptr.nsuccessor OF                   03206
                   = 0:   % null nsuccessor %                           03207
                      IF ptrx > 0                                       03208
                         THEN                                           03209
                            BEGIN                                       03210
                            &instptr ← ptrstk( ptrx ← ptrx-1 );
                                                                        03211
                            REPEAT CASE;                                03212
                            END                                         03213
                         ELSE                                           03214
                            REPEAT CASE 2 (cleanup);                    03215
                ENDCASE;                                                03216
                curptr.pmode ← parsing;                                 03217
                END;                                                    03218
             = backup:             % backup for repeat %                03219
                BEGIN                                                   03220
                &nextptr ← &curptr;                                     03221
                LOOP                                                    03222
                   BEGIN                                                03223
                   &instptr ← curptr.curnodeptr;                        03224
                   % $call the execution function in backup mode
                   if one exists %                                      03225
                      IF (&function ← curptr.pfunction ) # 0            03226
                         THEN function ( &curptr+$pathrecsize,
                         backup);                                       03227
                   % try to find an alternative to the current
                   inst %                                               03228
                      IF curptr.evalmod # $parallel                     03229
                         THEN                                           03230
                            BEGIN                                       03231
                            CASE &instptr ← instptr.alternative OF
                                                                        03232
                               = 0:                                     03233
                                  IF ptrx > curptr.ptrxsav THEN
                                                                        03234
                                     BEGIN                              03235
                                     &instptr ← ptrstk( ptrx ←
                                     ptrx-1);                           03236
                                     REPEAT CASE;                       03237
                                     END;                               03238
                               ENDCASE                                  03239
                                  EXIT LOOP;                            03240
                            END                                         03241
                         ELSE                                           03242
```

```
                        BEGIN                                   03243
                        % check to see if we are in a perform
                        loop %                                  03244
                           &instptr ← curptr.begnodeptr;        03245
                           IF &instptr = instptr.alternative
                           THEN EXIT LOOP;                      03246
                        END;                                    03247
                     % reset state counters %                   03248
                        ptrx ← curptr.ptrxsav;                  03249
                        evalx ← curptr.evalxsav;                03250
                     % current inst. has no alternative, so back up
                     path stack until we find a alternative.  If
                     none is found, then the command is terminated %
                                                                03251
                        IF (&curptr ← &curptr - $totalrecsize) <
                        &firstptr                               03252
                           THEN                                 03253
                              RETURN (FALSE); % parse failed %  03254
                     END;                                       03255
                     % setup to resume execution with current path
                     stack entry %                              03256
                        cueflg ← FALSE;                         03257
                        &pathptr ← 0;                           03258
                        &nextptr ← &curptr + $totalrecsize;     03259
                        % reset path stack index %              03260
                           pathx ← &nextptr - $pathstk;         03261
                        % reset eval stack index %              03262
                           evalx ← curptr.evalxsav;             03263
                        % reset feedback %                      03264
                           IF nlmode = fulldisplay THEN cfldsp();
                                                                03265
                        % resume execution %                    03266
                           GOTO resume;                         03267
                                                                03268
                  END;                                          03269
               = popselect:          % backup to inside of selection
               fuction %                                        03270
                  BEGIN                                         03271
                  % invoke the selection processor in pop mode.  We
                  pass only the resultptr and parsemode.  The rest
                  of the work will be done in the selection
                  processor %                                   03272
                     xselect( &curptr+$pathrecsize, popselect); 03273
                  % push the address of the result record onto the
                  eval stack %                                  03274
                     xpush( &curptr+$pathrecsize );             03275
                  curptr.pmode ← pnext;                         03276
                  END;                                          03277
            ENDCASE SIGNAL (interperr, $"Unrecognized Parse Mode");
                                                                03278
            END;                                                03279
      % we've screwed up if we get to here. %                   03280
         SIGNAL (interperr, $"Parser screwed up");              03281
      END.
                                                                03282
   (setbackup) PROC( % defines backup point after command completion
```

```
for repeat or insert mode command termination %                          0921
   % FORMAL ARGUMENTS %                                                  0922
      firstptr,  % ptr to first path stack entry %                       0923
      curptr);   % ptr to current path stack entry %                     0924
   % RETURNS %                                                           0925
      % ptr to path stack to which backup is to proceed %                0926
   LOCAL % VARIABLES %                                                   0927
      pathptr,   % ptr to head of path stack frame %                     0928
      ptr;       % ptr to path stack entry %                             0929
   REF ptr, firstptr, curptr, pathptr;                                   0930
   %------------%                                                        0931
   CASE complcode OF                                                     0932
      = 2:       % insert stmt %                                         0933
         BEGIN                                                          0934
         % check to make sure that the current system is the
         nlseditor, if not SIGNAL cmdelete which will cleanup as
         per CA character %                                             0935
            IF [$sbstack + sbstkx - $sbentsize].sbptr #
            $nlseditor                                                  0936
               THEN SIGNAL( cmdelete );                                 0937
         &ptr ← &firstptr;                                              0938
         END;                                                           0939
      = 3:       % rpt char %                                           0940
         BEGIN                                                          0941
         % rumble down path stack frames looking for one not
         beginning with a keyword recognition function %                0942
            &pathptr ← 0;                                               0943
            FOR &ptr ← &firstptr UP $totalrecsize UNTIL > &curptr
            DO                                                          0944
               BEGIN                                                    0945
               IF ptr.markptr # &pathptr THEN                           0946
                  &pathptr ← &ptr;                                      0947
               CASE [ptr.curnodeptr].opcode OF                          0948
                  IN [$confirm, $call]:                                 0949
                     RETURN( &pathptr );                                0950
                  ENDCASE;                                              0951
               END;                                                     0952
         % if we fall through to here, then the command can't be
         repeated, so signal the cmdelete %                             0953
            SIGNAL (cmdelete);                                          0954
         END;                                                           0955
      ENDCASE err(notyet);                                              0956
   RETURN( &ptr );                                                      0957
   END.                                                                 0958

(sleuth) PROCEDURE(                                                     02626
   % sleuth examines the current alternatives for the interpreter
   and sets up the action records to record what happens when one
   of the trigger characters is recognized  %                           02627
   % FORMAL ARGUMENTS %                                                 02628
      psptr,     % ptr to the current path stack record %              02629
      optptr,    % ptr to opt action record %                          02630
      captr,     % ptr to $pca action record %                         02631
      defaultptr);% ptr to default action ptr %                        02632
   % NORMAL RETURNS %                                                  02633
      % The action records for the opt action, $pca action, and
```

```
   default action are set up and completely filled in. %  02634
   % 1): the interpretation mode is returned  %           02635
      % = $parallel: $parallel recognition mode: %        02636
      % = $serial: $serial function execution mode %     02637
   % 2): a count of the number of alternative execution paths
   %                                                      02638
% ABNORMAL RETURNS %                                     02639
   % a signal is generated whenever an ambiguous construction
   in the grammar is detercted. %                         02640
LOCAL % VARIABLES %                                      02641
   op,          % opcode of instruction %                02642
   i,                % loop index variable %              02643
   brcount,   % number of branches in recognition seqence %
                                                          02644
   returnval,          % function return value %         02645
   headptr,   % ptr to head of alternative branch %      02646
   resetheader,% flag for resetting headptr %             02647
   instptr,   % ptr to instruction %                      02648
   lptrx,     % index of next location in lptrstk %       02649
   trivial,   % flag to prevent "trivial alternative"%   03304
   % lptrstk SHOULD BE ptrssize (in PDATA) WORDS LONG %   03760
      lptrstk[40]; % work stack for tree chasing %        02650
REF % VARIABLES %                                        02651
   headptr,   % ptr to head of nsuccessor list %         02652
   instptr,   % ptr to instruction %                      02653
   psptr,     % ptr to the current path stack record %    02654
   optptr,    % ptr to opt action record %               02655
   captr,     % ptr to $pca action record %              02656
   defaultptr;         % ptr to default action ptr %      02657
%-----------------%                                       02658
% initialize all fields of action records to default values %
                                                          02659
   optptr.propcode ← optptr.fbstrptr ← optptr.insptr ← 0; 02660
   captr.propcode ← captr.fbstrptr ← captr.insptr ← 0;    02661
   defaultptr.propcode ← defaultptr.fbstrptr ←
   defaultptr.insptr ← 0;                                 02662
trivial ← TRUE;                                          03305
% initialize local variables %                           02663
   &headptr ← &instptr ← psptr.curnodeptr;                02664
   resetheader ← FALSE;                                   02665
   lptrx ← 0;                                             02666
   returnval ← $parallel;                                 02667
   brcount ← 0;                                           02668
% we now set up actions to be accomplished whenever a trigger
character is encountered .  The characters currently
recognized as trigger characters are :                   02669
   1) CA character-- may be a bug selection.             02670
   2) option char-- indicated what to do when an $option is
   typed.                                                 02671
   3) any other char: --this is the default action for the
   type of recognition. %                                 02672
   % the action function is set to 0 if it is not permitted %
                                                          02673
% in order to set up all actions, we must check all
alternatives to the current instruction, as they will be
recognized in $parallel %                                02674
```

```
WHILE &instptr # 0 DO                                                   02675
   BEGIN                                                                02676
   CASE op ← instptr.opcode OF                                          02677
      = $keyop:    % keyword recognition %                              02678
         BEGIN                                                          02679
         BUMP brcount;                                                  02680
         CASE defaultptr.propcode OF                                    02681
            = 0:   % not yet defined %                                  02682
               BEGIN                                                    02683
               defaultptr.propcode ← op;                                02684
               defaultptr.insptr ← &headptr;                            02685
               defaultptr.fbstrptr ← $"0:";                             02686
               END;                                                     02687
            # op:                                                       02688
               SIGNAL (interperr, $"Ambiguous Grammar --
               multiple default actions ");                             02689
         ENDCASE;                                                       02690
         END;                                                           02691
      = $option,                                                        02692
      = $anyof:    % optional constructs %                             02693
         BEGIN                                                          02694
         CASE optptr.propcode OF                                        02695
            = 0:   % not yet defined %                                  02696
               BEGIN                                                    02697
               BUMP brcount;                                            02698
               optptr.propcode ← op;                                    02699
               optptr.insptr ← &headptr;                                02700
               optptr.fbstrptr ← $"[**]:";                              02701
               END;                                                     02702
            # op:                                                       02703
               SIGNAL (interperr, $"Ambiguous Grammar --
               multiple optional actions ");                            02704
         ENDCASE;                                                       02705
         resetheader ← TRUE;                                            02706
         REPEAT CASE (-1);                                              02707
         END;                                                           02708
      = $levadj:   % level adjust %                                     02709
         BEGIN                                                          02710
         CASE defaultptr.propcode OF                                    02711
            = 0:   % not yet defined %                                  02712
               BEGIN                                                    02713
               BUMP brcount;                                            02714
               defaultptr.propcode ← op;                                02715
               defaultptr.insptr ← &headptr;                            02716
               defaultptr.fbstrptr ← $"L:";                             02717
               END;                                                     02718
            # op:                                                       02719
               SIGNAL (interperr, $"Ambiguous Grammar --
               multiple default actions ");                             02720
         ENDCASE;                                                       02721
         CASE captr.propcode OF                                         02722
            = 0:   % not yet defined %                                  02723
               BEGIN                                                    02724
               captr.propcode ← op;                                     02725
               captr.insptr ← &headptr;                                 02726
               captr.fbstrptr ← $"L:";                                  02727
```

```
                END;                                                    02728
              # op:                                                     02729
                SIGNAL (interperr, $"Ambiguous Grammar --
                multiple $pca actions ");                               02730
            ENDCASE;                                                    02731
            END;                                                        02732
          = $vwspecs:  % viewspecs %                                    02733
            BEGIN                                                       02734
            CASE defaultptr.propcode OF                                 02735
              = 0:   % not yet defined %                                02736
                BEGIN                                                   02737
                BUMP brcount;                                           02738
                defaultptr.propcode ← op;                               02739
                defaultptr.insptr ← &headptr;                           02740
                defaultptr.fbstrptr ← $"V:";                            02741
                END;                                                    02742
              # op:                                                     02743
                SIGNAL (interperr, $"Ambiguous Grammar --
                multiple default actions ");                            02744
            ENDCASE;                                                    02745
            CASE captr.propcode OF                                      02746
              = 0:   % not yet defined %                                02747
                BEGIN                                                   02748
                captr.propcode ← op;                                    02749
                captr.insptr ← &headptr;                                02750
                captr.fbstrptr ← $"V:";                                 02751
                END;                                                    02752
              # op:                                                     02753
                SIGNAL (interperr, $"Ambiguous Grammar --
                multiple $pca actions ");                               02754
            ENDCASE;                                                    02755
            END;                                                        02756
          = $confirm:  % command confirmations %                        02757
            CASE captr.propcode OF                                      02758
              = 0:   % not yet defined %                                02759
                BEGIN                                                   02760
                BUMP brcount;                                           02761
                captr.propcode ← op;                                    02762
                captr.insptr ← &headptr;                                02763
                captr.fbstrptr ← $"OK:";                                02764
                END;                                                    02765
              # op:                                                     02766
                SIGNAL (interperr, $"Ambiguous Grammar --
                multiple $pca actions ");                               02767
            ENDCASE;                                                    02768
          = $dsel:                                                      02769
            BEGIN                                                       02770
            BUMP brcount;                                               02771
            CASE defaultptr.propcode OF                                 02796
              = 0:   % not yet defined %                                02797
                BEGIN                                                   02798
                defaultptr.propcode ← $getdae;                          02799
                defaultptr.insptr ← &headptr;                           02800
                defaultptr.fbstrptr ← $"A:";                            02801
                END;                                                    02802
              ENDCASE                                                   02803
```

```
                SIGNAL (interperr, $"Ambiguous Grammar --
                multiple default actions ");                    02804
            IF nlmode = fulldisplay THEN                        02772
               BEGIN                                            02774
               CASE captr.propcode OF                           02775
                  = 0:        % not yet defined %               02776
                     BEGIN                                      02777
                     captr.propcode ← $getbug;                  02778
                     captr.insptr ← &headptr;                   02779
                     captr.fbstrptr ← $"B:";                    02780
                     END;                                       02781
                  ENDCASE                                       02782
                     SIGNAL (interperr, $"Ambiguous Grammar --
                     multiple $pca actions ");                  02783
               END                                              02793
            ELSE                                                02794
               BEGIN                                            02795
               CASE captr.propcode OF                           02805
                  = 0:        % not yet defined %               02806
                     BEGIN                                      02807
                     captr.propcode ← $getdae;                  02808
                     captr.insptr ← &headptr;                   02809
                     captr.fbstrptr ← $"A:";                    02810
                     END;                                       02811
                  ENDCASE                                       02812
                     SIGNAL (interperr, $"Ambiguous Grammar --
                     multiple ca actions ");                    02813
               END;                                             02814
            END;                                                02815
         = $ssel:                                               02816
            BEGIN                                               02817
            BUMP brcount;                                       02818
            CASE defaultptr.propcode OF                         02852
               = 0:   % not yet defined %                       02853
                  BEGIN                                         02854
                  defaultptr.propcode ← $getdae;                02855
                  defaultptr.insptr ← &headptr;                 02856
                  defaultptr.fbstrptr ← $"A:";                  02857
                  END;                                          02858
               ENDCASE                                          02859
                  SIGNAL (interperr, $"Ambiguous Grammar --
                  multiple default actions ");                  02860
            CASE optptr.propcode OF                             02870
               = 0:   % not yet defined %                       02871
                  BEGIN                                         02872
                  optptr.propcode ← $getlit;                    02873
                  optptr.insptr ← &headptr;                     02874
                  optptr.fbstrptr ← $"[T]:";                    02875
                  END;                                          02876
               ENDCASE                                          02877
                  SIGNAL (interperr, $"Ambiguous Grammar --
                  multiple optional actions ");                 02878
            IF nlmode = fulldisplay THEN                        02819
               BEGIN                                            02821
               CASE captr.propcode OF                           02831
                  = 0:        % not yet defined %               02832
```

```
                    BEGIN                                          02833
                    captr.propcode ← $getbug;                      02834
                    captr.insptr ← &headptr;                       02835
                    captr.fbstrptr ← $"B:";                        02836
                    END;                                           02837
                 ENDCASE                                           02838
                    SIGNAL (interperr, $"Ambiguous Grammar --
                    multiple $pca actions ");                      02839
              END                                                  02849
           ELSE                                                    02850
              BEGIN                                                02851
              CASE captr.propcode OF                               02861
                 = 0:        % not yet defined %                   02862
                    BEGIN                                          02863
                    captr.propcode ← $getdae;                      02864
                    captr.insptr ← &headptr;                       02865
                    captr.fbstrptr ← $"A:";                        02866
                    END;                                           02867
                 ENDCASE                                           02868
                    SIGNAL (interperr, $"Ambiguous Grammar --
                    multiple $pca actions ");                      02869
              END;                                                 02879
           END;                                                    02880
        = $lsel:                                                   02881
           BEGIN                                                   02882
           BUMP brcount;                                           02884
           CASE defaultptr.propcode OF                             02885
              = 0:   % not yet defined %                           02886
                 BEGIN                                             02887
                 defaultptr.propcode ← $getlit;                    02888
                 defaultptr.insptr ← &headptr;                     02889
                 defaultptr.fbstrptr ← $"T:";                      02890
                 END;                                              02891
              ENDCASE                                              02892
                 SIGNAL (interperr, $"Ambiguous Grammar --
                 multiple default actions ");                      02893
           CASE optptr.propcode OF                                 02903
              = 0:   % not yet defined %                           02904
                 BEGIN                                             02905
                 optptr.propcode ← $getdae;                        02906
                 optptr.insptr ← &headptr;                         02907
                 optptr.fbstrptr ← $"[A]:";                        02908
                 END;                                              02909
              ENDCASE                                              02910
                 SIGNAL (interperr, $"Ambiguous Grammar --
                 multiple optional actions ");                     02911
           IF nlmode = fulldisplay THEN                            04950
              BEGIN                                                04951
              CASE captr.propcode OF                               04952
                 = 0:        % not yet defined %                   04953
                    BEGIN                                          04954
                    captr.propcode ← $getbug;                      04955
                    captr.insptr ← &headptr;                       04956
                    captr.fbstrptr ← $"B:";                        04957
                    END;                                           04958
                 ENDCASE                                           04959
```

```
                    SIGNAL (interperr, $"Ambiguous Grammar --
                    multiple $pca actions ");                    04960
                END                                              04961
             ELSE                                                04962
                BEGIN                                            04963
                CASE captr.propcode OF                           04964
                   = 0:        % not yet defined %               04974
                      BEGIN                                      04975
                      captr.propcode ← $getlit;                  04976
                      captr.insptr ← &headptr;                   04977
                      captr.fbstrptr ← $"T:";                    04978
                      END;                                       04979
                   ENDCASE                                       04971
                      SIGNAL (interperr, $"Ambiguous Grammar --
                      multiple $pca actions ");                  04972
                END;                                             04973
             END;                                                02912
          = $xecute:                                             02913
             BEGIN                                               02914
             % stack the instptr into the lptrstk %              02915
                lptrstk[ lptrx ] ← &instptr;                     02916
                IF (lptrx ← lptrx+1) > $ptrssize                 02917
                   THEN SIGNAL (interperr, $"lptrstk
                   overflowed");                                 02918
             % set up new instptr and headptr values %           02919
                &headptr ← &instptr ← instptr.addr;              02920
             REPEAT CASE;                                        02921
             END;                                                02922
          = $call:     % arbitrary execution function %         02923
             BEGIN                                               03311
             % assume that this is the first alternative and
             has already "failed" as there is no way to predict
             what this one does %                                02924
             trivial ← FALSE;                                    03306
             IF returnval = $parallel THEN returnval ← $serial;
                                                                 02925
             END;                                                03310
          = $prcall:   % parsing function %                      02926
             BEGIN                                               03309
             IF returnval = $parallel THEN returnval ←
             $parsefunction;                                     02927
             trivial ← FALSE;                                    03307
             END;                                                03308
          ENDCASE                                                02928
             BEGIN                                               02929
             IF instptr.opcode = $option THEN                    02930
                CASE &instptr ← instptr.nsuccessor OF            02931
                   = 0:                                          02932
                      IF lptrx > 0 THEN                          02933
                         BEGIN                                   02934
                         &instptr ← lptrstk[lptrx ← lptrx-1];
                                                                 02935
                         REPEAT CASE;                            02936
                         END                                     02937
                      ELSE SIGNAL (interperr, $"Meaningless
                      grammar");                                 02938
```

```
                        ENDCASE                                          02939
                           BEGIN                                         02940
                           IF resetheader := FALSE THEN                  02941
                              &headptr ← &instptr;                       02942
                           REPEAT CASE 2;                                02943
                           END;                                          02944
                     % ***** have a "trivial alternative" *** (often a
                     mistake in the grammar) ***** lets execute it and
                     hope the writer of the cml program will fix up his
                     code when he can't reach more meaningful
                     alternatives %                                      02945
                     IF trivial THEN                                     02951
                        BEGIN                                            02952
                        psptr.curncdeptr ← &instptr;                     02953
                        FOR i ← 0 UP UNTIL >= lptrx DO                   02954
                           BEGIN                                         02955
                           ptrstk[ptrx] ← lptrstk[i];                    02956
                           BUMP ptrx;                                    02957
                           END;                                          02958
                        RETURN( $serial, 1 );                            02959
                        END;                                             02960
                     END;                                                02961
               % get the next alternative (of the instptr) %             02962
                  CASE &headptr ← instptr.alternative OF                 02963
                     = 0:                                                02964
                        IF lptrx > 0 THEN                                02965
                           BEGIN                                         02966
                           &instptr ← lptrstk[lptrx ← lptrx-1];          02967
                           REPEAT CASE;                                  02968
                           END;                                          02969
                     = &instptr: % check for loop condition %            02970
                        BEGIN                                            02971
                        &headptr ← 0;                                    02972
                        REPEAT CASE (0);                                 02973
                        END;                                             02974
                  ENDCASE;                                               02975
               % set headptr to point to the head of the alternative
               chain %                                                   02976
                  &instptr ← &headptr;                                   02977
               END;                                                      02978
         RETURN (returnval, brcount);                                    02979
         END.
                                                                         02980
      (evaler) PROCEDURE( % sets up for and interprets an instruction %
                                                                         01306
         % FORMAL ARGUMENTS %                                            01307
            curptr);   % ptr to path stack entry %                       01308
         % NORMAL RETURNS %                                              01309
            % none %                                                     01310
         % ABNORMAL RETURNS %                                            01311
         LOCAL % VARIABLES %                                             01312
            count,              % count of arguments processed %          01313
            function,           % ptr to execution function %             01314
            instptr,            % ptr to the current instruction %        01315
            op,                 % opcode of current inst %                01316
            options,            % count of alternatives %                 01317
```

```
   proc,                % ptr to $pfcall parsing function % 01318
   stateptr,            % ptr to the function state record %
                                                           01319
   lptrx,     % index of next location in lptrstk %        06229
   % lptrstk SHOULD BE ptrssize (in PDATA) WORDS LONG %    06230
      lptrstk[40], % work stack for tree chasing %         06231
   tempptr,             % misc temp. ptr %                 01320
   arg[8];              % argument collection vector %     01321
LOCAL TEXT POINTER tp1, tp2;                               04945
% REF VARIABLES %                                         03803
   REF fbstr, inpt;                                        03804
   REF tda, sysmsg, nlpcmdstk;                             03805
   REF                                                     01322
      curptr,                                              01323
      function,                                            01324
      instptr,                % ptr to the current instruction
      %                                                    01325
      proc,                                                01326
      stateptr,               % ptr to the function state
      record %                                             01327
      tempptr;                                             01328
LOCAL STRING                                               01329
   helpstr[25], work[10];     % prompts string for $pfcall
   functions %                                             01330
%-----------------------%                                  01331
(again): % come here after optional construct is taken %   01332
% initialize local variables %                            01333
   &instptr ← curptr.curnodeptr;                           01334
   op ← instptr.opcode;                                    01335
% interrogate the alternatives to find out how to interpret
the next instruction if the next instruction has alternatives
%                                                          01336
IF NOT cueflg AND ( instptr.alternative OR op IN
[$keyop,$execute]) THEN                                    01337
   CASE curptr.evalmod ← sleuth( &curptr, $optaction,
   $caaction, $defaction :options ) OF                     01338
      = $parallel:    % recognizer may be invoked %        01339
         BEGIN                                             01340
         % prompt the user if appropriate %                01341
            % put together a feedback string for prompting %
                                                           01342
               edistr(  $prompstr );                       01343
         % set up for parameter recognition %              01344
            fbctl( startparams );                          01345
         % look to see what is comin# next if required %   01346
            IF options > 1 THEN                            01347
               BEGIN                                       01348
               % output the feedback string %             01349
                  IF inprompts # noprompts AND NOT cueflg THEN
                                                           01350
                     fbctl( incues, $prompstr );           01351
               cueflg ← TRUE;                              01352
               CASE lookc() OF                             01353
                  = rptchar:                               01354
                     BEGIN                                 01355
                     IF pathx > $totalrecsize              01356
```

```
                THEN GOTO cacharcase;                    01357
             inpt(); % read the char %                   01358
             % return last cmd chars to input buffer %
                                                         01359
                resetb( $keyrptstr );                    01360
             REPEAT CASE;                                01361
             END;                                        01362
          = inschar:                                     01363
             BEGIN                                       01364
             IF pathx > $totalrecsize                    01365
                THEN GOTO cacharcase;                    01366
             % check to make sure that the current
             system is the nlseditor, %                  01367
                IF ($sbstack + sbstkx -
                $sbentsize).sbptr # $nlseditor           01368
                   THEN GOTO cacharcase;                 01369
             inpt(); % read the char %                   01370
             &instptr ← $zinsstatement;                  01371
             cueflg ← FALSE;                             01372
             END;                                        01373
          = cacnar:                                      01374
             (cacharcase):                               01375
             IF (&instptr ← caaction.insptr) = 0         01376
                THEN                                     01377
                   BEGIN                                 01378
                   inpt();                               01379
                   fbctl( '? );                          01380
                   REPEAT CASE;                          01381
                   END;                                  01382
          = optchar:                                     01383
             IF (&instptr ← optaction.insptr) = 0        01384
                THEN                                     01385
                   BEGIN                                 01386
                   inpt();                               01387
                   fbctl( '? );                          01388
                   REPEAT CASE;                          01389
                   END                                   01390
                ELSE                                     01391
                   BEGIN                                 01392
                   cueflg ← FALSE;                       01393
                   IF optaction.propcode IN
                   (#option,#anyof)                      01394
                      THEN                               01395
                         BEGIN                           01396
                         inpt();                         01397
                         % romp down the current path
                         stacking ptrs to execute
                         instructions if any %           01398
                         &tempptr ← curptr.curnodeptr;
                                                         01399
                         WHILE tempptr.opcode =
                         #xecute DO                      01400
                         BEGIN                           01401
                         % stack ptr to execute inst
                         into ptrstk %                   01402
                         ptrstk[ ptrx ] ← &tempptr;
```

```
                                                                01403
                        IF (ptrx ← ptrx + 1) >
                        $ptrssize                               01404
                           THEN SIGNAL (interperr,
                           $"ptrstk overflowed");
                                                                01405
                        &tempptr ← tempptr.addr;                01406
                        END;                                    01407
                        % stack ptr to optional inst
                        into ptrstk %                           01408
                           ptrstk[ ptrx ] ← &instptr;
                                                                01409
                           IF (ptrx ← ptrx + 1) >
                           $ptrssize                            01410
                           THEN SIGNAL (interperr,
                           $"ptrstk overflowed");
                                                                01411
                        % evaluate the optional inst.
                        %                                       01412
                           curptr.curnodeptr ←
                           instptr.addr;                        01413
                        GOTO again;                             01414
                        END;                                    01415
                  END;                                          01416
      = BC, =BW:                                                01417
         BEGIN                                                  01418
         inpt();                                                01419
         SIGNAL (popstate);                                     01420
         END;                                                   01421
      = CD:                                                     01422
         BEGIN                                                  01423
         inpt();                                                01424
         SIGNAL (cmdelete);                                     01425
         END;                                                   01426
      = '?:                                                     01427
         BEGIN                                                  01428
         inpt();                                                01429
         &fbstr ← $"";                                          01430
         fbhelp( &curptr, cmdmode );                            01431
         REPEAT CASE;                                           01432
         END;                                                   01433
      = 'S-100B:  % <↑S> %                                      06897
         BEGIN                                                  06898
         inpt();                                                06899
         &fbstr ← $ \\;                                          06900
         cshelp( &curptr, cmdmode, TRUE );                      06901
         REPEAT CASE;                                           06902
         END;                                                   06903
   ENDCASE                                                      01434
      IF (&instptr ← defaction.insptr) = 0                      01435
         THEN                                                   01436
           BEGIN                                                01437
           inpt();                                              01438
           fbctl( '? );                                         01439
           REPEAT CASE;                                         01440
           END;                                                 01441
```

```
            END;                                                    01442
         % update the curnodeptr field in the path stack
         record %                                                   01443
            curptr.curnodeptr ← &instptr;                           01444
            op ← instptr.opcode;                                    01445
         END;                                                       01446
      = $serial:       % execution function %                      01447
         NULL;                                                      01448
      = $parsefunction:        % parsing function %                01449
         BEGIN                                                      01450
         % romp down the current path stacking ptrs to execute
         instructions if any %                                      01451
            &instptr ← curptr.curnodeptr;                           01452
            WHILE instptr.opcode = $xecute DO                       01453
               BEGIN                                                01454
               % stack ptr to execute inst into ptrstk %            01455
               ptrstk[ ptrx ] ← &instptr;                           01456
               IF (ptrx ← ptrx + 1) > $ptrssize                     01457
                  THEN SIGNAL (interperr, $"ptrstk
                  overflowed");                                     01458
               &instptr ← instptr.addr;                             01459
               END;                                                 01460
            curptr.curnodeptr ← &instptr;                           01461
         % build prompt string for builtin functions %              01462
            % put together a feedback string for prompting %
                                                                    01463
            edistr( $prompstr );                                    01464
         % call parse functions in "parsehelp" mode to solicit
         a subhelp string %                                         01465
            lptrx ← 0;                                              06328
            *helpstr* ← NULL;                                       01467
            WHILE &instptr DO                                       05937
               BEGIN                                                05938
               CASE op ← instptr.opcode OF                          05939
                  = $xecute:                                        06160
                     BEGIN                                          06161
                     % stack the instptr into the lptrstk %
                                                                    06162
                        lptrstk[ lptrx ] ← &instptr;                06163
                        IF (lptrx ← lptrx+1) > $ptrssize            06164
                           THEN SIGNAL (interperr, $"lptrstk
                           overflowed");                            06165
                     % set up new instptr values %                 06166
                        &instptr ← instptr.addr;                    06167
                     REPEAT CASE;                                   06168
                     END;                                           06169
                  = $pfcall:  % parsing function %                  06176
                     BEGIN                                          06177
                     &proc ← instptr.addr;                          01466
                     *work* ← NULL;                                 06329
                     proc( &curptr, parsehelp, $work );             01468
                     IF work.L THEN                                 06228
                        IF helpstr.L THEN                           06330
                           *helpstr* ← *helpstr*, '/, *work*
                                                                    06227
                        ELSE                                        06331
```

```
                        *helpstr* ← *work*;                    06332
                    END;                                       06180
                = $option :                                    06183
                    CASE &instptr ← instptr.nsuccessor OF
                                                               06184
                        = 0:                                   06185
                            IF lptrx > 0 THEN                  06186
                                BEGIN                          06187
                                &instptr ← lptrstk[lptrx ←
                                lptrx-1];                      06188
                                REPEAT CASE;                   06189
                                END                            06190
                            ELSE SIGNAL (interperr,
                            $"Meaningless grammar");           06191
                        ENDCASE                                06192
                            BEGIN                              06193
                            REPEAT CASE 2;                     06196
                            END;                               06197
                    ENDCASE;                                   06325
                % get the next alternative (of the instptr) %
                                                               06210
                    CASE instptr.alternative OF                06211
                        = 0:                                   06212
                            IF lptrx > 0 THEN                  06213
                                BEGIN                          06214
                                &instptr ← lptrstk[lptrx ←
                                lptrx-1];                      06215
                                REPEAT CASE;                   06216
                                END                            06217
                            ELSE &instptr ← 0;                 06327
                        = &instptr: % check for loop condition %
                                                               06218
                            BEGIN                              06219
                            REPEAT CASE (0);                   06221
                            END;                               06222
                        ENDCASE &instptr ← instptr.alternative;
                                                               06223
                END;                                           06226
            &instptr ← curptr.curnodeptr;                      06326
        % append helpstr in front of the prompts string %
                                                               01469
            IF helpstr.L > 0 THEN                              01470
                BEGIN                                          04936
                LOOP                                           04938
                    BEGIN                                      04939
                    IF NOT                                     04940
                        FIND SF(*helpstr*) [':] ↑tp1 ↑tp2 ←tp1
                        THEN EXIT LOOP;                        04942
                    *helpstr* ←                                04941
                        SF(*helpstr*) tp1, tp2 SE(*helpstr*);
                                                               04943
                    END;                                       04944
                IF prompstr.L > 0 THEN                         01471
                    *prompstr* ← *helpstr*, '/, *prompstr*     01472
                ELSE *prompstr* ← *helpstr*, ':;               01473
                END;                                           04937
```

```
            % output the prompt %                                   01474
               fbctl( incues, $prompstr );                          01475
               cueflg ← TRUE;                                       07150
            % check next inpt char for trigger action %             01476
               CASE lookc() OF                                      01477
                  = '?:                                             01478
                     BEGIN                                          01479
                     inpt();                                        01480
                     &fbstr ← $"";                                  01481
                     fbhelp( &curptr, cmdmode );                    01482
                     REPEAT CASE;                                   01483
                     END;                                           01484
                  = 'S-100B:  % <↑S> %                              06904
                     BEGIN                                          06905
                     inpt();                                        06906
                     &fbstr ← $"";                                  06907
                     cshelp( &curptr, cmdmode, TRUE );              06908
                     REPEAT CASE;                                   06909
                     END;                                           06910
                  = BC, = BW:                                       01485
                     BEGIN                                          03312
                     inpt();                                        03313
                     SIGNAL( popstate );                            01486
                     END;                                           03316
                  = CD;                                             01487
                     BEGIN                                          03314
                     inpt();                                        03315
                     SIGNAL( cmdelete );                            01486
                     END;                                           03317
                  ENDCASE;                                          01489
            END;                                                    01490
         ENDCASE SIGNAL (interperr, $"Unexpected evalmode
         value");                                                   01491
      % decode the current node in the grammar %                    01492
         % the decoding process sets up the path stack record
         (curptr) for evaluation by the functional execution
         process.  Some of the "builtin" processes are transformed
         into function execution processes for calls on external
         routines %                                                 01493
         decode(&curptr);                                           01494
      % process the current node %                                  01495
         % if the pfunction field of the path stack record is not
         NULL, then the arguments are collected and the function is
         invoked %                                                  01496
         IF (&function ← curptr.pfunction) # 0                      01497
            THEN                                                    01498
               BEGIN                                                01499
               % collect ptrs to arguments to arg vector %          01500
                  FOR count ← curptr.argcount DOWN UNTIL < 1 DO     01501
                     arg[count] ← xpop();                           01502
               % set the stateptr to point to the function state
               record %                                             01503
                  &stateptr ← &curptr + $pathrecsize;               01504
               % $call processing routine, pushing ptr to current
               node onto eval stack if the routine returns TRUE %
                                                                    01505
```

```
            IF function(&stateptr, curptr.pmode, arg[1], arg[2],
            arg[3], arg[4], arg[5], arg[6], arg[7], arg[8])          01506
               THEN                                                  01507
                  BEGIN                                              01508
                  curptr.pmode ← pnext;                              01509
                  IF op NOT IN [$fbclear, $store] THEN               01510
                     xpush( &stateptr );                             01511
                  % clear cueflg if the instruction just executed
                  was a recognizer %                                 01512
                     IF op IN [$keyop,$call]                         01513
                        THEN cueflg ← FALSE;                         01514
                  END                                                01515
               ELSE                                                  01516
                  curptr.pmode ← backup;                             01517
            END                                                      01518
         ELSE curptr.pmode ← pnext;                                  01519
   RETURN;                                                           01520
   END.
                                                                     01521
(testselect) PROCEDURE(                                              01522
   % FUNCTION %                                                      01523
      % this routine checks to see if the path stack entry
      indicated by curptr points to a selection function which
      can be backed up. %                                            01524
   % FORMAL ARGUMENTS %                                             01525
      curptr);   % ptr to path stack entry %                        01526
   % RETURNS %                                                       01527
      % 1) TRUE/FALSE  boolean condition %                          01528
   LOCAL % VARIABLES %                                               01529
      statesavptr;        % ptr to selection state info %           01530
   REF % VARIABLES %                                                 01531
      statesavptr,        % ptr to selection state info %           01532
      curptr;    % ptr to path stack entry %                        01533
   %-------------%                                                   01534
   IF [curptr.curnodeptr].opcode IN [$ssel, $lsel] THEN              01535
      BEGIN                                                          01536
      &statesavptr ← &curptr + $pathrecsize + pselLen;               01537
      IF statesavptr.nselects > 0 THEN                               01538
         BEGIN                                                       01539
         curptr.pmode ← popselect;                                   01540
         ptrx ← curptr.ptrxsav;                                      01541
         evalx ← curptr.evalxsav - 1;                                01542
         RETURN (TRUE);                                              01543
         END;                                                        01544
      END;                                                           01545
   RETURN (FALSE);                                                   01546
   END.                                                              01547
(decode) PROCEDURE(                                                  01548
   % FUNCTION %                                                      01549
      % decode decodes the interpretive instruction indicated by
      the curnodeptr field of the path stack entry indicated by
      psptr and either executes the instruction if it is simple
      enough or sets up the path stack entry for subsequent
      function invocation %                                          01550
      % the opcodes processed locally require no saving of state
```

```
   information and no processing during backup operations and
   include the following list: ENTER, LOAD, STORE, VALUEOF%
                                                                01551
% FORMAL ARGUMENTS %                                            01552
   psptr);     % ptr to path stack entry %                       01553
% NORMAL RETURNS %                                              01554
   % none %                                                     01555
% ABNORMAL RETURNS %                                            01556
   % SIGNALS ARE GENERATED AS FOLLOWS: %                        01557
      % 1) interperr -- interpreter error %                     01558
LOCAL % VARIABLES %                                             01559
   args,       % number of arguments %                         01560
   functaddr,          % address of the processing function %
                                                                01561
   funstateptr,        % ptr to function state record %         01562
   instptr,    % pointer to current instruction %              01563
   lastptr,    % ptr to previous path stack entry %            01564
   op,         % interpreter function code %                   01565
   temp,       % temporary value %                             01566
   tempptr,    % scratch pointer %                             01567
   val;        % value/address field of interpreter word %     01568
REF % VARIABLES %                                               01569
   funstateptr,                                                 01570
   instptr,                                                     01571
   lastptr,                                                     01572
   psptr,                                                       01573
   tempptr;                                                     01574
%-----------------------%                                       01575
% initialize local variables %                                 01576
   args ← functaddr ← 0;                                        01577
   &funstateptr ← &psptr + $pathrecsize;                        01578
(over):                                                         01579
% strip apart the instruction %                                01580
   &instptr ← psptr.currnodeptr;                                01581
   op ← instptr.opcode;                                         01582
   val ← instptr.addr;                                          01583
% process by function code %                                   01584
   CASE op OF                                                   01585
      % RECOGNIZERS %                                          01586
         = $keyop:     % keyword recognition operator %         01587
            BEGIN                                               01588
            % construct a single argument whose value is the
            pointer to the path stack entry %                   01589
               xpush( &psptr );                                 01590
               args ← 1;                                        01591
               functaddr ← $keywrec;                            01592
            END;                                                01593
         = $dsel,      % destination selection %                01594
         = $lsel,      % literal selection %                    01595
         = $ssel:      % source selection %                     01596
            BEGIN                                               01597
            args ← 2;                                           01598
            functaddr ← $xselect;                               01599
            xpush( &psptr );                                    01600
            % set up maxselects and nselects:%                  01601
               &tempptr ← &funstateptr + 4;                     01602
```

```
                tempptr.maxselects ← tempptr.nselects ← 0;
            END;
        = $vwspecs:  % gets viewspecs %
            BEGIN
            needconfirm ← FALSE;
            functaddr ← $xviewspecs;
            END;
        = $levadj:   % get level adjust string %
            BEGIN
            needconfirm ← FALSE;
            functaddr ← $xlevadj;
            END;
        = $confirm:  % get command confirmation %
            BEGIN
            functaddr ← $xconfirm;
            % argument is opcode of previous instruction %
                xpush( lastsel );
                args ← 1;
            END;
        = $anyof:
            BEGIN
            % look to see if $anyof node is at the top of the
            eval stack, if not, then we must initialize an
            entry at the top of the stack %
                &lastptr ← xread() - $pathrecsize;
                IF [lastptr.curnodeptr].opcode # $anyof THEN
                    BEGIN
                    xpush( &funstateptr );
                    funstateptr ← 0;
                    END;
            END;
        = $option:
            NULL;
    % CONTROL ELEMENTS %
        = $pfcall:   % parse function $call %
            BEGIN
            functaddr ← val;
            args ← instptr.val2;
            END;
        = $xecute:   % transfer to another point in tree %
            BEGIN
            IF defaction.propcode =  $keyop
                THEN REPEAT CASE ($keyop);
            % stack ptr to $xecute inst in ptrstk and evaluate
            the descendent path %
                ptrstk[ ptrx ] ← &instptr;
                IF (ptrx ← ptrx + 1) > $ptrssize
                    THEN SIGNAL (interperr, $"ptrstk
                    overflowed");
                psptr.curnodeptr ← instptr.addr;
                GOTO over;
            END;
        = $call:     % subroutine $call %
```

```
                BEGIN                                                   01652
                % provide running feedback if in DNLS and user
                wants feedback %                                        01653
                   IF nlmode = fulldisplay                              01654
                      AND inprompts # noprompts THEN                    01655
                         fbctl( incues, $"...");                        01656
                functaddr ← val;                                        01657
                args ← instptr.val2;                                    01658
                END;                                                    01659
            % FEEDBACK ELEMENTS %                                       01660
              = $fbclear:  % clear feedback buffer %                    01661
                BEGIN                                                   01662
                % set address to point to interface routine %           01663
                   functaddr ← $xfeedback;                              01664
                % push arguments onto eval stack %                      01665
                   xpush( $fbclear );                                   01666
                   args ← 1;                                            01667
                END;                                                    01668
              = $necho:     % echo noise word string %                   01669
                BEGIN                                                   01670
                % set address to point to interface routine %           01671
                   functaddr ← $xfeedback;                              01672
                % push arguments onto eval stack %                      01673
                   xpush( $necho );                                     01674
                   xpush( val );                                        01675
                   args ← 2;                                            01676
                END;                                                    01677
              = $recho:     % replace last thing echoed %               01678
                BEGIN                                                   01679
                % set address to point to interface routine %           01680
                   functaddr ← $xfeedback;                              01681
                % push arguments onto eval stack %                      01682
                   xpush( $recho );                                     01683
                   xpush( val );                                        01684
                   args ← 2;                                            01685
                END;                                                    01686
            % VALUE MANIPULATIONS %                                     01687
              = $pload:     % $pload variable to ptr stack %           01688
                BEGIN                                                   01689
                temp ← [val]; % fetch contents of variable %            01690
                % check to make sure that the frame pointed to by
                the rh of temp still has a fcounter value which =
                the left half of temp %                                 01691
                   IF temp.LH # [temp.RH - $pathrecsize].fcounter
                                                                        01692
                      THEN SIGNAL ( interperr, $"reference to
                      undefined interpreter variable");                 01693
                % push the ptr to the value record onto the eval
                stack %                                                 01694
                   xpush( temp.RH );                                    01695
                END;                                                    01696
              = $store:     % $store value into variable %              01697
                BEGIN                                                   01698
                functaddr ← $xstor;                                     01699
                args ← 1;                                               01700
                xpush( val );                                           01701
```

```
                END;                                                    01702
            = Senter:    % Senter constant into stack %                 01703
                BEGIN                                                   01704
                xpush( &funstateptr );                                  01705
                funstateptr.alword ← val;                               01706
                END;                                                    01707
            ENDCASE SIGNAL(interperr, $"unrecognizable interpreter op
            code");                                                     01708
        % terminate processing, return to caller %                      01709
            psptr.pfunction ← functaddr;                                01710
            psptr.argcount ← args;                                      01711
            RETURN;                                                     01712
        END.
                                                                        01713
% EVAL STACK MANIPULATION ROUTINES %                                    01714
    (xpush) PROCEDURE(                                                  01715
        % pushes the address of an argument record onto eval stack %
                                                                        01716
        % FORMAL ARGUMENTS %                                            01717
            addrofvalue ); % address of value %                         01718
        % NORMAL RETURNS %                                              01719
            % none %                                                    01720
        % ABNORMAL RETURNS %                                            01721
            % SIGNAL interperr generated for eval stack overflow %      01722
        %---------------------%                                         01723
        IF evalx NOT IN [0,$evalsize)                                   01724
            THEN SIGNAL (interperr, $"Eval stack out of range");        01725
        evalstk[ evalx ] ← addrofvalue;                                 01726
        BUMP evalx;                                                     01727
        RETURN;                                                         01728
        END.
                                                                        01729
    (xpop) PROCEDURE;                                                   01730
        % pops the address of an argument record from the eval stack %
                                                                        01731
        % FORMAL ARGUMENTS %                                            01732
            % none %                                                    01733
        % NORMAL RETURNS %                                              01734
            % address of argument record contained in the top of the
            eval stack %                                                01735
        % ABNORMAL RETURNS %                                            01736
            % SIGNAL interperr generated for eval stack underflow %
                                                                        01737
        %---------------------%                                         01738
        IF evalx NOT IN (0,$evalsize]                                   01739
            THEN SIGNAL (interperr, $"Eval stack out of range");        01740
        evalx ← evalx - 1;                                              01741
        RETURN (evalstk[evalx]);                                        01742
        END.
                                                                        01743
    (xstore) PROCEDURE(                                                 01744
        % stores the address of an argument record onto eval stack %
                                                                        01745
        % FORMAL ARGUMENTS %                                            01746
            addrofvalue ); % address of value %                         01747
        % NORMAL RETURNS %                                              01748
```

```
      % none %                                                    01749
    % ABNORMAL RETURNS %                                         01750
      % SIGNAL interperr generated for eval stack out of range %
                                                                  01751
    %----------------------%                                     01752
    IF evalx NOT IN (0,$evalsize]                               01753
      THEN SIGNAL (interperr, $"Eval stack out of range");      01754
    evalstk[evalx-1] ← addrofvalue;                             01755
    RETURN;                                                     01756
    END.
                                                                01757
  (xread) PROCEDURE;                                            01758
    % reads the address of an argument record from the eval stack
    %                                                           01759
    % FORMAL ARGUMENTS %                                        01760
      % none %                                                  01761
    % NORMAL RETURNS %                                         01762
      % address of argument record contained in the top of the
      eval stack %                                              01763
    % ABNORMAL RETURNS %                                       01764
      % SIGNAL interperr generated for eval stack out of range %
                                                                01765
    %----------------------%                                     01766
    IF evalx NOT IN (0,$evalsize]                               01767
      THEN SIGNAL (interperr, $"Eval stack out of range");      01768
    RETURN (evalstk[evalx-1]);                                  01769
    END.
                                                                01770
% I/O SUPPORT ROUTINES %                                        01771
  (edistr) PROCEDURE( % edits terminal prompt string %          01772
    % FORMAL ARGUMENTS %                                       01773
      targetptr);% ptr to result string %                       01774
    % RETURNS %                                                 01775
      % none %                                                  01776
    % ABNORMAL RETURNS %                                       01777
      % none %                                                  01778
    LOCAL % VARIABLES %                                         01779
      i,                    % loop counter %                   01780
      sourceptr;            % ptr to source string %           01781
    LOCAL TEXT POINTER tp1, tp2;                                04926
    REF % VARIABLES %                                           01782
      sourceptr,            % ptr to argument string %         01783
      targetptr;            % ptr to result string %           01784
    %--------------------%                                      01785
    *targetptr* ← NULL;                                         01786
    % set sourceptr to alternatively point to each of the three
    possible prompt strings, and add these strings to the target
    string if appropriate %                                     01787
      FOR i ← 1 UP UNTIL > 3 DO                                 01788
        BEGIN                                                   01789
        &sourceptr ← CASE i OF                                  01790
          = 1: caaction.fbstrptr;                               01792
          = 2: defaction.fbstrptr;                              01791
          = 3: optaction.fbstrptr;                              01793
          ENDCASE 0;                                            01794
        IF &sourceptr # 0 THEN                                  01795
```

```
                BEGIN                                                   01796
                % exit if we are ignoring optional prompts and the
                prompt string contains a $option character (*) %        01797
                   IF inprompts = partprompts                           01798
                      AND (FIND SF(*sourceptr*) ['[])                   01799
                         THEN REPEAT LOOP;                              01800
                CASE targetptr.L OF                                     01801
                   > 0:                                                 01802
                      IF NOT FIND SF(*targetptr*) [*sourceptr*]         01803
                         THEN *targetptr* ← *targetptr*, "/",
                         *sourceptr*                                    01804
                   = 0:                                                 01805
                      *targetptr* ← *sourceptr*;                        01806
                ENDCASE;                                                01807
                LOOP                                                    04927
                   BEGIN                                                04928
                   IF NOT                                               04929
                      FIND SF(*targetptr*) [':] ↑tp1 ↑tp2 ←tp1          04930
                      THEN EXIT LOOP;                                   04931
                   *targetptr* ← SF(*targetptr*) tp1, tp2
                   SE(*targetptr*);                                     04932
                   END;                                                 04933
                *targetptr* ← *targetptr*, ':;                          04934
                END;                                                    01808
             END;                                                       01809
          RETURN;                                                       01810
          END.
                                                                        01811
% BUILTIN INTERPRETER FUNCTIONS %                                       01812
   (xfeedback) PROCEDURE(                                               01813
      % provide feedback elements for CML %                             01814
      % FORMAL ARGUMENTS %                                              01815
         resultptr,          % ptr to the result record %               01816
         parsemode,          % command parsing mode %                   01817
         fbmode,     % feedback mode %                                  01818
         fbstring );         % pointer to feedback string %             01819
      % NORMAL RETURNS %                                                01820
         % 1) returns the result ptr %                                  01821
      % ABNORMAL RETURNS %                                              01822
         % generates a SIGNAL if the feedback mode is not recognized
         %                                                              01823
      REF % VARIABLES %                                                 01824
         resultptr,                                                     01825
         fbstring;                                                      01826
      %-----------------%                                               01827
      CASE parsemode OF                                                 01828
         = parsing:                                                     01829
            BEGIN                                                       01830
            % save the current length of the feedback buffer in the
            function state record (pointed to by resultptr) %           01831
               resultptr[1] ← cflstr.L;                                 01832
            CASE fbmode OF                                              01833
               = $necho:                                                01834
                  BEGIN                                                 01835
                  fbctl( echostr, &fbstring );                          01836
                  END;                                                  01837
```

```
                = $recho:                                                   01838
                   BEGIN                                                    01839
                   fbctl( rechostr, &fbstring );                            01840
                   END;                                                     01841
                = $fbclear:                                                 01842
                   BEGIN                                                    01843
                   fbctl( clearcfl );                                       01844
                   END;                                                     01845
             ENDCASE SIGNAL (interperr, $"unrecognized feedback code
             in xfeedback");                                                01846
             END;                                                           01847
          = cleanup,                                                        01848
          = backup:                                                         01849
             BEGIN                                                          01850
             % restore the command feedback line length %                   01851
                cflstr.L ← resultptr[1];                                    01852
             END;                                                           01853
       ENDCASE;                                                             01854
       RETURN (&resultptr);                                                 01855
       END.
                                                                            01856
    (xstor) PROCEDURE(                                                      01857
       % perform the $store operation %                                     01858
       % FORMAL ARGUMENTS %                                                01859
          resultptr,          % ptr to the result record %                  01860
          parsemode,          % command parsing mode %                      01861
          location);          % ptr to $store location %                    01862
       % NORMAL RETURNS %                                                  01863
          % 1) returns the result ptr %                                     01864
       LOCAL % VARIABLES %                                                  01865
          frameptr;  % ptr to current frame %                               01866
       REF % VARIABLES %                                                    01867
          frameptr,                                                         01868
          resultptr,                                                        01869
          location;                                                         01870
       %-----------------%                                                 01871
       CASE parsemode OF                                                    01872
          = parsing:                                                        01873
             BEGIN                                                          01874
             % fetch the record pointer contained in the top of the
             eval stack, save this away in the function state record
             (for backup purposes) %                                        01875
                resultptr ← xpop();                                         01876
             % set frameptr to point to the path stack record
             associated with that frame %                                   01877
                &frameptr ← resultptr - $pathrecsize;                       01878
             % pack the frame counter for the frame with the address
             of the result record %                                         01879
                resultptr.LH ← frameptr.fcounter;                           01880
             % save the augmented ptr away in the variable %               01881
                location ← resultptr;                                       01882
             END;                                                           01883
          = cleanup,                                                        01884
          = backup:                                                         01885
             BEGIN                                                          01886
             % push the ptr to the saved frame back onto the eval
```

```
            stack %                                                    01887
               % set ptr to frame %                                    01888
                  &frameptr ← &resultptr - $pathrecsize;               01889
               % set evalx so eval stack will look like it did
               before the $store operation was executed %              01890
                  evalx ← frameptr.evalxsav - 1;                       01891
               % push ptr to result record back onto eval stack %
                                                                       01892
                  xpush( resultptr.RH );                               01893
            END;                                                       01894
         ENDCASE;                                                      01895
         RETURN (&resultptr);                                          01896
         END.
                                                                       01897
% KEYWORD RECOGNITION AND FEEDBACK SUPPORT %                          01898
   % KEYWORD RECOGNITION ROUTINE %                                    01899
      (keywrec) PROC(                                                  06647
         % FORMAL ARGUMENTS %                                         06648
            resultptr,          % ptr to result record %              06649
            parsemode,          % parsing mode %                      06650
            curptr );  % ptr to path stack entry %                    06651
         % FUNCTION %                                                 06652
            % this routine performs all recognition and feedback
            functions for identifying a command keyword.  Four
            recognition methods are presently supported:
               1) recognition of some subset of the keyword list by an
            initial character
               2) recognition of all elements of the keyword list by a
            minimim unique initial substring.                          06653
               3) mdemand recognition requiring a right delimiter.     06654
               4) recognition based upon a fixed number of characters.
                                                                       06655
            The second recognition mode is entered if either an initial
            left delimiter (SP) is encountered.  A question mark may be
            typed at any point, and the keywords still available for
            recognition are printed out for the user. %                06656
         % NORMAL RETURNS %                                            06657
            % this routine returns the vocabulary index of the
            recognized keyword.  If recognition fails, a value of 0 is
            returned.  If the nofail $option is selected (nofail =
            TRUE), then control remains in this routine until some
            valid keyword is recognized %                              06658
         % ABNORMAL RETURNS %                                          06659
            % recognition of CD causes a GOTO STATE %                  06660
            % typing BC or Bw when the feedback buffer is empty is
            interpretted as a request to back up the parse, and a
            signal named "popstate" is generated %                     06661
         LOCAL                                                         06662
            list,       % ptr to head of keyword list %               06663
            recogflag,          % TRUE if have recognized keyword %    06664
            fullflag,  % TRUE if alternate recognition is emplyed %
                                                                       06665
            echotype,  % type of echoing after recognition %          06666
            cmode,     % local copy of cmdmode %                      06667
            recmode,   % working recog. mode (recogmode/recog2mode) %
                                                                       06668
```

```
    nextchar,  % next inpt character %                          06669
    hits,      % number of compares %                           06670
    hindex,    % hit index into vocab of string %               06671
    hitx,      % temp hit index %                               06672
    oldptrx,   % starting value for ptrx %                      06673
    oldptrx,   % starting value for ptrx %                      06673
    % work SHOULD BE wasize (in PDATA) WORDS LONG %             06674
       work[30];       % state buffer for keyinit and nextkey %
                                                                06675
```

```
                                                                06675
REF                                                             06676
   list,                                                        06677
   curptr,                                                      06678
   resultptr;                                                   06679
REF fbstr, inpt;                                                06680
REF tda, sysmsg, hlpcmdstk;                                     06681
%----------------------%                                        06682
kwrstate ← FALSE;                                               06683
CASE parsemode OF                                               06684
   = parsing:                                                   06685
      BEGIN                                                     06686
      % set up some state info (change later) %                 06687
         &list ← curptr.curnodeptr;                             06688
         oldptrx ← ptrx;                                        06689
         recogflag ← fullflag ←  FALSE;                         06690
         recmode ← recogmode;                                   06691
         echotype ← 1; % full echoing of keyword %              06692
         cmdmode ← 0;                                           06693
         IF nlmode                                              06694
            THEN cmdmode.dnlscmd ← TRUE                         06695
            ELSE cmdmode.tnlscmd ← TRUE;                        06696
         IF recmode = mexpert                                   06697
            THEN cmdmode.llcmd ← TRUE;                          06698
         cmode ← cmdmode;                                       06699
         % set up fbstr in 5th and subsequent words of work
         area %                                                 06700
            &fbstr ← $work + 4;                                 06701
            fbstr.M ← $fbstrmax; fbstr.L ← 0;                   06702
         % save away current state values %                     06703
            resultptr.keyinlength ← keyinpstr.L; % save
            current length of keyword save string %             06704
            resultptr.ksaveflag ← keysaveflag;                  06705
            resultptr.fblen ← cflstr.L;                         06706
         % perfom any global functions associated with keyword
         recognition %                                          06707
            fbctl( startrec );                                  06708
         % prompt the user if not already done so %             06709
            IF NOT cueflg THEN fbctl( incues, $"C:" );          06710
      % read next character and look for control characters %
                                                                06711
         CASE nextchar ← inpcuc() OF                            06712
            =SP:      % left/right delimiter character %        06713
               BEGIN                                            06714
               kwrstate ← 1;                                    06715
```

```
= mexpert:                                                    06717
   % check for use as a left delimiter %
                                                              06718
```

```
                    IF fbstr.L = 0 AND NOT fullflag                    06719
                       THEN                                            06720
                          BEGIN %  start of alternate
                          recognition %                                06721
                          IF nlmode NOT= fulldisplay THEN
                          typech(SP);                                  06722
                          fullflag ← TRUE;                             06723
                          cmode.llcmd ← FALSE;                         06724
                          recmode ← recog2mode;                        06725
                          REPEAT CASE 2;                               06726
                          END;                                         06727
              = mdemand:                                               06728
                 % check for a right delimiter %                       06729
                    IF recogflag                                       06730
                       THEN                                            06731
                          BEGIN                                        06732
                          echotype ← 2; % conditional echo
                          %                                            06733
                          REPEAT CASE 2 (-1); % EXIT
                          routine %                                    06734
                          END;                                         06735
           ENDCASE;                                                    06736
           % fall through if not a delimiter character %
                                                                       06737
              REPEAT CASE (0);                                         06738
           END;                                                        06739
        =CD:                                                           06740
           SIGNAL (cmdelete);                                          06741
        =BC:                                                           06742
           BEGIN                                                       06743
           IF fullflag AND (recogflag = 0) THEN                        06744
              BEGIN                                                    06745
              fullflag ← FALSE;                                        06746
              cmode.llcmd ← TRUE;                                      06747
              recmode ← recogmode;                                     06748
              recogflag ← 0;                                           06749
              END                                                      06750
           ELSE                                                        06751
              BEGIN                                                    06752
              recogflag ← MAX(0,recogflag-1);                          06753
              fbctl( nextchar );                                       06754
              END;                                                     06755
           REPEAT CASE;                                                06756
           END;                                                        06757
        =BW:                                                           06758
           BEGIN                                                       06759
           recogflag ← FALSE;                                          06760
           fullflag ← FALSE;                                           06761
           cmode.llcmd ← TRUE;                                         06762
           recmode ← recogmode;                                        06763
           fbctl( nextchar );                                          06764
           REPEAT CASE;                                                06765
           END;                                                        06766
        ='?:       % request for subhelp %                             06767
           BEGIN                                                       06768
           fbhelp( &curptr, cmode);                                    06769
```

```
    REPEAT CASE;                                                    06770
    END;                                                            06771
= 'S-100B:  %  <↑S> request for syntax subhelp %
                                                                    06911
    BEGIN                                                           06912
    cshelp( &curptr, cmode, TRUE);                                  06913
    REPEAT CASE;                                                    06914
    END;                                                            06915
= ALT,                                                              06772
= $eschar:          % possible right delimiter %
                                                                    06773
    BEGIN                                                           06774
    kwrstate ← 1;                                                   06775
    % check for a right delimiter %                                 06776
       IF recmode = mdemand AND recogflag AND
       fbstr.L # 0                                                  06777
          THEN                                                      06778
             BEGIN                                                  06779
             % EXIT the routine %                                   06780
                REPEAT CASE  (-1);                                  06781
             END;                                                   06782
    % fall through to ENDCASE %                                    06783
       REPEAT CASE (0);                                             06784
    END;                                                            06785
= -1:     % common routine exit point %                            06786
    BEGIN                                                           06787
    kwrstate ← 1;                                                   06788
    % set ptrx to value set up by nextkey %                        06789
       ptrx ← work.savex;                                           06790
       ptrstk[ptrx-1] ← work.savstk;                                06791
    resultptr ← hindex;                                             06792
    IF hindex.LH # 0                                                06793
       THEN resultptr ← hindex.LH;                                  06794
    % save input chars for possible repeat cmd %
                                                                    06795
       IF keysaveflag THEN                                          06796
          BEGIN                                                     06797
          IF fbstr.L > 0 THEN                                       06798
             BEGIN                                                  06799
             IF fullflag % excape from expert %
                                                                    06800
                THEN *keyinpstr* ← *keyinpstr*, SP;
                                                                    06801
             *keyinpstr* ← *keyinpstr*, *fbstr*;
                                                                    06802
             END;                                                   06803
          IF (recmode = mexpert AND NOT fullflag)
                                                                    06804
             OR recmode = mdemand THEN                              06805
                *keyinpstr* ← *keyinpstr*,
                nextchar;                                           06806
          END;                                                      06807
    % echo the keyword string %                                    06808
       CASE echotype OF                                             06809
          = 1: % full feedback %                                    06810
             fbctl( keyword, hindex.RH );                           06811
```

```
                = 2: % conditional feedback %                    06812
                   IF nlmode = fulldisplay                       06813
                      OR fbackmode = verbsmode                   06814
                         THEN REPEAT CASE(1)                     06815
                         ELSE fbctl( addchar, SP);               06816
                ENDCASE;                                         06817
             kwrstate ← -1;                                      06818
             RETURN( &resultptr );                               06819
             END;                                                06820
          ENDCASE                                                06821
             BEGIN                                               06822
             kwrstate ← 1;                                       06823
             % initialize the keyword search %                   06824
                ptrx ← cldptrx;                                  06825
                keyinit( &curptr, $work, cmode, FALSE );
                                                                 06826
             IF cmode.llcmd                                      06827
                THEN                                             06828
                   BEGIN % single character recognition mode
                   %                                             06829
                   IF hindex ← nextkey( $work, nextchar )
                                                                 06830
                      THEN                                       06831
                         BEGIN                                   06832
                         REPEAT CASE (-1); % EXIT %              06833
                         END                                     06834
                      ELSE                                       06835
                         % didn't find anything appropriate
                         %                                       06836
                         IF nofail THEN                          06837
                             % put up ? and try again %          06838
                            BEGIN                                06839
                            fbctl( '? );                         06840
                            REPEAT CASE;                         06841
                            END;                                 06842
                   END                                           06843
                ELSE                                             06844
                   BEGIN % multiple character recognition
                   mode %                                        06845
                   *fbstr* ← *fbstr*, nextchar;                  06846
                   hits ← 0;                                     06847
                   WHILE (hitx ← nextkey( $work, nextchar ))
                   # 0 DO                                        06848
                      IF (hits ← hits + 1) = 1 THEN              06849
                         BEGIN                                   06850
                         hindex ← hitx;                          06851
                         fbctl( addchar, nextchar);              06852
                         END                                     06853
                      ELSE REPEAT CASE;                          06854
                   % hits is the count of partial matches %
                                                                 06855
                      CASE hits OF                               06856
                         =1:        % have found the keyword
                         %                                       06857
                            BEGIN                                06858
                            BUMP recogflag;                      06859
```

```
                                    CASE recmode OF                          06860
                                       = mexpert,                            06861
                                       = manticipatory:                      06862
                                          BEGIN                              06863
                                          REPEAT CASE 3 (-1); % exit
                                          %                                  06864
                                          END;                               06865
                                       = mfixed:                             06866
                                          IF fbstr.L = fix1 OR
                                          fbstr.L = [hindex.RH].L
                                          THEN                               06867
                                          REPEAT CASE
                                          (manticipatory);                   06868
                                    ENDCASE;                                 06869
                                    REPEAT CASE 2;                           06870
                                    END;                                     06871
                                 =0:        % no matches %                   06872
                                    BEGIN                                    06873
                                    fbstr.L ← fbstr.L-1; % get rid
                                    of the bum char from the
                                    feedback buffer %                        06874
                                    fbctl( '? );                             06875
                                    END;                                     06876
                                 ENDCASE;                                    06877
                              IF nofail THEN REPEAT CASE;                    06878
                           END;                                              06879
                     END;                                                    06880
                  ptrx ← oldptrx;                                            06881
                  kwrstate ← FALSE;                                          06882
                  RETURN (0); % FAILURE EXIT %                               06883
               END;                                                          06884
            = backup,                                                        06885
            = cleanup:                                                       06886
               BEGIN                                                         06887
               % reset feedback state information %                          06888
                  cflstr.L ← resultptr.fblen;                                06889
               % reset keyword input save info (for possible repeat
               cmd) %                                                        06890
                  keyinpstr.L ← resultptr.keyinlength;                       06891
                  keysaveflag ← resultptr.ksaveflag;                         06892
               END;                                                          06893
         ENDCASE;                                                            06894
         RETURN;                                                             06895
         END.

                                                                             06896
% RECOGNITION SUPPORT ROUTINES %                                            02126
   (vocab) PROC( index, astr ); %returns vocabulary string, given
   index and address of astring %                                            02127
      LOCAL                                                                  02128
         fflag,                                                              05171
         bp,         % byte pointer %                                         02129
         count,      % character count in vocab string %                      02130
         char;       % next char in vocabulary string %                       02131
      REF astr;                                                              02132
      %----------------------%                                               02133
      *astr* ← NULL;                                                         02134
```

```
    count ← 0;                                                          02135
    bp ← chbmty + index;                                                02136
    UNTIL (char ← ↑bp) = 0 DO                                           02137
       BEGIN                                                            02138
       IF (count ← count + 1) # 1 THEN                                  02139
          BEGIN                                                         05172
          IF fflag THEN                                                 05175
             CASE char OF                                               02140
                IN ['A, 'Z]:                                            02141
                   char ← char + 40B; % force to lower case %           02142
             ENDCASE;                                                   02143
          END                                                           05173
       ELSE fflag ← IF char = '< THEN FALSE ELSE TRUE;                  05174
       *astr* ← *astr*, char;                                           02144
       END;                                                             02145
    RETURN;                                                             02146
    END.
                                                                        02147
(nextkey) PROCEDURE( work, nextchar );                                  02148
    LOCAL                                                               02149
       bp,          % byte pointer %                                    02150
       bp2,         % byte pointer %                                    02151
       cmode,       % local copy of work.currcm %                       02152
       curptr,      % ptr to current path stack %                       02153
       i,                  % index variable for compare loop %          02154
       instptr,     % ptr to current instruction word %                 02155
       l,                  % current length of recognition string %
                                                                        02156
       proc,        % $pfcall function address %                        02157
       result,      % return result %                                   02158
       tempptr,     % temp inst. ptr %                                  02159
       value;       % compare value for multi char search %             02160
    REF                                                                 02161
       curptr,                                                          02162
       instptr,                                                         02163
       proc,                                                            02164
       tempptr,                                                         02165
       work;                                                            02166
    REF fbstr, inpt;                                                    05169
    REF tda, sysmsg, nlpcmdstk;                                         05170
    %----------------------%                                            02167
    % set up some state info (change later) %                          02168
       cmode ← work.currcm;                                             02169
       result ← 0;                                                      02170
       l ← fbstr.L;                                                     02171
    % chase down the data structures looking for a match %             02172
       WHILE (&instptr ← work.currinst) # 0 AND result = 0 DO           02173
          BEGIN                                                         02174
          % decrement work.optcount if endopt was set on the last
          time through here %                                           02175
          IF work.endopt                                                02176
             THEN                                                       02177
                BEGIN                                                   02178
                work.optcount ← work.optcount - 1;                      02179
                work.endopt ← FALSE;                                    02180
                END;                                                    02181
```

```
        CASE instptr.opcode OF                                          02182
          = $keyop:                                                     02183
            BEGIN                                                       02184
            IF work.optflag AND work.optcount = 0                       02185
               THEN EXIT CASE;                                          02186
            IF instptr.ctrl .A cmode = cmode                            02187
               THEN                                                     02188
                  IF cmode.llcmd THEN                                   02189
                     BEGIN                                              02190
                     bp ← chbmty + instptr.addr;                        02191
                     IF nextchar < 0 OR ↑bp = nextchar                  02192
                        THEN REPEAT CASE (-1);                          02193
                     END                                                02194
                  ELSE                                                  02195
                     BEGIN                                              02196
                     % if in secondary expert mode dont check
                     Ll cmds%                                           02197
                        IF instptr.ctrl.llcmd AND work.usrmode
                        = mexpert
                         THEN EXIT CASE;                                02198
                     % exit if string is empty %                        02199
                        IF l = 0 THEN REPEAT CASE (-1);                 02200
                     bp2 ← chbmty + &fbstr;                             02201
                     bp ← chbmty + instptr.addr;                        02202
                     IF l <= 5 THEN % do fast compare %                 02203
                        BEGIN                                           02204
                        bp.bpsize ← bp2.bpsize ← 7 * l;                 02205
                        value ← ↑bp2;                                   02206
                        IF ↑bp = value THEN REPEAT CASE (-1);
                                                                        02207
                        END                                             02208
                     ELSE                                               02209
                        BEGIN % compare each character %                02210
                        FOR i ← 1 UP UNTIL > l DO                       02211
                           IF ↑bp # ↑bp2 THEN EXIT CASE; % no
                           hit %                                        02212
                           REPEAT CASE (-1); % hit %                    02213
                        END;                                            02214
                     END;                                               02215
            END;                                                        02216
          = $xecute:                                                    02217
            BEGIN                                                       02218
            % stack ptr to the instruction and repeat the case
            with the invoked tree as our next alternative %
                                                                        02219
               ptrstk[ ptrx ] ← &instptr;                               02220
               BUMP ptrx;                                               02221
               IF (work.nextx ← ptrx) > $ptrssize                       02222
                  THEN SIGNAL (interperr, $"ptrstk
                  overflowed");                                         02223
               &instptr ← work.currinst ← instptr.addr;                 02224
               REPEAT CASE;                                             02225
            END;                                                        02226
          = $option,                                                    02227
          = $anyof:                                                     02228
            IF nextchar < 0 % for subhelp purposes only %               02229
```

```
                OR work.optflag % recognition of optional keywords
                %                                                   02230
                   THEN                                             02231
                      BEGIN                                         02232
                      work.optcount ← work.optcount + 1;            02233
                      work.firstinst ← &instptr;                    02234
                      REPEAT CASE  ($xecute);                       02235
                      END;                                          02236
             = -1: % come here when we've found a partial match %
                                                                    02237
                BEGIN                                               02238
                &curptr ← work.currpath;                            02239
                curptr.curnodeptr ←                                 02240
                   IF work.optcount > 0                             02241
                      THEN work.firstinst                           02242
                      ELSE &instptr;                                02243
                result ← instptr.addr;                              02244
                result.LH ← instptr.val2;                           02245
                work.savex ← ptrx;                                  02246
                work.savstk ← ptrstk[ptrx-1];                       02247
                END;                                                02248
             = $confirm:                                            02249
                IF nextchar < 0                                     02250
                   THEN result ← $"<OK>";                           02251
             = $ssel,                                               02252
             = $lsel,                                               02253
             = $dsel:                                               02254
                IF nextchar < 0                                     02255
                   THEN result ← $"<CONTENT>";                      02256
             = $vwspecs:                                            02257
                IF nextchar < 0                                     02258
                   THEN result ← $"<VIEWSPECS>";                    02259
             = $levadj:                                             02260
                IF nextchar < 0                                     02261
                   THEN result ← $"<LEVEL-ADJUST>";                 02262
             = $pfcall:                                             02263
                IF nextchar < 0 THEN                                02264
                   BEGIN                                            02265
                   &curptr ← work.currpath;                         02266
                   &proc ← instptr.addr;                            02267
                   *savhelpstr* ← NULL;                             02268
                   % $call the proc in "parsehelp" mode to find
                   out what it is doing %                           02269
                      proc( &curptr, parsehelp, $savhelpstr );
                                                                    02270
                   result ← $savhelpstr;                            02271
                   END;                                             02272
          ENDCASE                                                   02273
             IF nextchar < 0 THEN                                   02274
                BEGIN % look for some printable "subhelp"
                information %                                       02275
                &tempptr ← instptr.nsuccessor;                      02276
                WHILE &tempptr # 0 DO                               02277
                   CASE tempptr.opcode OF                           02278
                      = $keyop,                                     02279
                      = $xecute,                                    02280
```

```
                        = $option,                                      02281
                        = $anyof:                                       02282
                           BEGIN                                        02283
                           &instptr ← &tempptr;                         02284
                           REPEAT CASE 2;                               02285
                           END;                                         02286
                        IN [$confirm, $levadj]:                         02287
                           REPEAT CASE 2 (tempptr.opcode);              02288
                     ENDCASE                                            02289
                        &tempptr ← tempptr.nsuccessor;                  02290
                  END;                                                  02291
            % chase down alternatives to current instruction, if
            there are no more alternatives, then the ptrstk is
            popped to get a next instruction %                          02292
               CASE &tempptr ← instptr.alternative OF                   02293
                  = 0:      % null alternative chain %                  02294
                     IF work.nextx > work.firstx THEN                   02295
                        BEGIN                                           02296
                        BUMP DOWN ptrx;                                 02297
                        &instptr ← ptrstk[ work.nextx ← ptrx];          02298
                        % if leaving an optional construct, then set
                        the $option flag %                              02299
                           CASE instptr.opcode OF                       02300
                              = $option,                                02301
                              = $anyof:                                 02302
                                 work.endopt ← TRUE;                    02303
                           ENDCASE;                                     02304
                        REPEAT CASE;                                    02305
                        END;                                            02306
                  = &instptr:       % check for a looping construct
                  %                                                     02307
                     REPEAT CASE (&tempptr ← 0);                        02308
               ENDCASE;                                                 02309
            work.currinst ← &instptr ← &tempptr;                        02310
            END;                                                        02311
      RETURN (result);                                                  02312
      END.
                                                                        02313
(keyinit) PROCEDURE( curptr, work, lcmdmode, mode);                     05251
      % keyinit sets up a static work record for use by nextkey to
      sequence through the alternative keyword lists %                  05252
      LOCAL instptr;                                                    05253
      REF instptr, curptr, work;                                        05254
      REF fbstr, inpt;                                                  05255
      REF tda, sysmsg, nlpcmdstk;                                       05256
      %----------------------%                                          05257
      % set up some state info (change later) %                        05258
         &instptr ← curptr.begncdeptr;                                  05259
         IF NOT mode AND instptr.opcode IN [$option, $anyof]            05260
            THEN &instptr ← instptr.addr;                               05261
         work.firstinst ← work.currinst ← &instptr;                     05262
         work.currcm ← lcmdmode;                                        05263
         work.optflag ← FALSE;                                          05264
         work.endopt ← FALSE;                                           05265
         work.optcount ← 0;                                             05266
         work.firstx ← work.nextx ← ptrx;                               05267
```

```
        work.currpath ← &curptr;                                        05268
        work.usrmode ←                                                  05269
           (IF lcmdmode.llcmd THEN recogmode ELSE recog2mode);          05270
     RETURN;                                                            05271
     END.
                                                                        05272
% FEEDBACK CONTROL ROUTINES %                                          02334
   (fbctl) PROCEDURE( function, pl ); % feedback control routine %
                                                                        04351
     % user feedback operations are performed here,  above this
     level there should not be much need for knowing whether the
     user is in TNLS or DNLS %                                          04352
     LOCAL                                                              04353
        ls,         % ptr to link stack entry %                       04354
        lsl,        % ptr to link stack entry %                       04355
        char,       % character value %                               04356
        l;                  % length count %                          04357
     LOCAL STRING tempstr[100];                                         04358
     % REF VARIABLES %                                                 04359
        REF fbstr, inpt;                                                04360
        REF tda, sysmsg, hlpcmdstk;                                     04361
        REF                                                             04362
           lsl,     % ptr to link stack entry %                       04363
           ls;      % ptr to link stack entry %                       04364
     %-------------------------%                                       04365
     l ← fbstr.L;                                                       04366
     IF nlmode = fulldisplay THEN                                       04367
        BEGIN % DNLS %                                                  04368
        CASE function OF                                                04369
           =clearcfl:               % clear feedback buffer %          04370
              BEGIN                                                     04371
              *fbstr* ← NULL;                                           04372
              cflstr.L ← cflpos ← 0;                                    04373
              cfldsp();                                                 04374
              disarm();                                                 04375
              END;                                                      04376
           =addchar:        % add character to feedback buffer %
                                                                        04377
              BEGIN                                                     04378
              IF l > 1                                                  04379
                 THEN DSP(...,*fbstr* )                                 04380
                 ELSE DSP( *fbstr* );                                   04381
              END;                                                      04382
           =keyword:        % feedback keyword %                       04383
              BEGIN                                                     04384
              vocab( pl, &fbstr );                                      04385
              IF l > 0                                                  04386
                 THEN DSP(...,*fbstr* )                                 04387
                 ELSE DSP( *fbstr* );                                   04388
              *fbstr* ← NULL;                                           04389
              IF fbackmode = tersemode ∝then
                 dsp[↑);                                                04391
              END;                                                      04392
           =startrec:       % begin keyword recognition %              04393
              BEGIN                                                     04394
              DSP(↑);                                                   05160
```

```
   END;                                                          04396
=BC:            % backspace character %                        04397
   IF 1 <= 0                                                     04398
      THEN REPEAT CASE (BW)                                      04399
      ELSE                                                       04400
         BEGIN % delete last character ffrom string %            04401
         fbstr.L ← 1 - 1;                                        04402
         DSP(...*fbstr*  );                                      04403
         END;                                                    04404
=BW:            % backspace word %                             04405
   IF 1 > 0 THEN                                                 04406
      BEGIN                                                      04407
      *fbstr* ← NULL;                                            04408
      cflstr.L ← MAX( 0, cflpos-1 );                             04409
      cfldsp();                                                  04410
      END                                                        04411
   ELSE SIGNAL (popstate); % back up to previous point
   in parse %                                                    04412
='?:            % unrecognizable command %                     04413
   BEGIN                                                         04414
   IF auxinflag % reading from aux. input source %               04415
      THEN auxinterminate();                                     04416
   qm();        % put out question mark %                       04417
   lookc();     % wait for next character %                     04418
   END;                                                          04419
=echostr:       % echo supplied string in cfl %                04420
   IF fbackmode = verbsmode THEN                                 04421
      BEGIN                                                      04422
      *tempstr* ← '(, *[pl]*, ');                                04423
      DSP( *tempstr*  );                                         04424
      END;                                                       04425
=rechostr:      % replace previously supplied string in
cfl %                                                            04426
   BEGIN                                                         04427
   *tempstr* ← '(, *[pl]*, ');                                   04428
   DSP( ...*tempstr*  );                                         04429
   END;                                                          04430
=fbendlit:      % append to lit str and wait for input %
                                                                 04431
   BEGIN                                                         04432
   aplit( pl );                                                  04433
   lookc(); % wait for user to type in next char %               04434
   rstlit();                                                     04435
   END;                                                          04436
=typelit:       % put up literal string (not in cfl) %
                                                                 04437
   BEGIN                                                         04438
   litdpy( pl );                                                 04439
   lookc(); % wait for user to type in next char %               04440
   rstlit();                                                     04441
   END;                                                          04442
=typenullit:    % put up null lit string (not in cfl) %
                                                                 04443
   BEGIN                                                         04444
   litdpy( $"" );                                                04445
```

```
            END;                                                    04446
         =typecalit:     % put up literal string (not in cfl) %
                                                                    04447
            BEGIN                                                   04448
            litdpy( p1 );                                           04449
            REPEAT CASE(addcalit);                                  05928
            END;                                                    04454
         =addcalit:      % put up literal string (not in cfl) %
                                                                    04455
            BEGIN                                                   04456
            aplit($"

            Type <CA> to continue.");                               04457
            % wait for user to type in CA or CD %                   04458
               CASE inpt() OF                                       05929
                  = CA: rstlit();                                   05930
                  = CD:                                             05931
                     BEGIN                                          05933
                     rstlit();                                      05932
                     SIGNAL(statesig);                              05935
                     END;                                           05934
                  ENDCASE REPEAT CASE;                              05936
            END;                                                    0446
         =startparams:   % begin parameter collection %             04462
            BEGIN                                                   04467
            IF NOT &tda THEN &tda ← lda();        % for tabs %
                                                                    04464
            IF fbackmode = tersemode                                04465
               THEN af();                                           04466
            END;                                                    04467
         =incues:         % provide inpt prompts %                  04468
            IF inprompt # noprompts THEN                            04469
               BEGIN                                                04470
               *cflarw* ← " > ", *[p1]*;                            04471
               arowon ← TRUE;                                       04472
               mrk();                                               04473
               END;                                                 04474
         =fbpop:                 % cleanup after popping states %
                                                                    04475
            BEGIN                                                   04476
            cueflg ← FALSE;                                         04477
            cfldsp();                                               04478
            END;                                                    04479
         =fbaddlit:      % add p1 to literal area %                 04480
            BEGIN                                                   04481
            aplit( p1 );                                            04482
            IF nldevice = devlproc THEN track();                    04483
            END;                                                    04484
         ENDCASE err($"Illegal call to fbctl");                     04485
      END                                                           04486
   ELSE                                                             04487
      BEGIN % TNLS %                                                04488
      CASE function OF                                              04489
         =clearcfl:              % clear feedback buffer %          04490
            BEGIN                                                   04491
            *fbstr* ← NULL;                                         04492
```

```
      crlf();                                                    04493
      END;                                                       04494
   =addchar:        % echo character from feedback buffer %
                                                                 04495
      BEGIN                                                      04496
      IF l > 1 THEN                                              04497
         CASE p1 OF                                              04498
            IN ['A, 'Z]:                                         04499
               p1 ← p1 + 40B; % force to lower case %            04500
         ENDCASE;                                                04501
      todco( p1 );                                               04502
      END;                                                       04503
   =keyword:        % feedback keyword %                        04504
      BEGIN                                                      04505
      IF l < feedbk THEN                                         04506
         BEGIN                                                   04507
         vocab( p1, &fbstr );                                    04508
         *fbstr* ←                                               04509
            *fbstr*[1+1 TO MIN( fbstr.L, feedbk)];               04510
         echo( &fbstr );                                         04511
         END;                                                    04512
      todco( SP );                                               04513
      *fbstr* ← NULL;                                            04514
      END;                                                       04515
   =startrec:       % begin keyword recognition %               04516
      NULL;                                                      04517
   =BC:             % backspace character %                     04518
      IF l <= 0                                                  04519
         THEN REPEAT CASE (BW)                                   04520
         ELSE                                                    04521
            BEGIN % delete last character ffrom string v
                                                                 04522
            todco( '\ );                                         04523
            todco( *fbstr*[ l ] );                               04524
            fbstr.L ← l - 1;                                     04525
            END;                                                 04526
   =BW:             % backspace word %                          04527
      IF l > 0 THEN                                              04528
         BEGIN                                                   04529
         *fbstr* ← NULL;                                         04530
         todco( '← );                                            04531
         END                                                     04532
      ELSE SIGNAL (popstate); % back up to previous point
      in parse %                                                 04533
   ='?:             % unrecognizable command %                  04534
      BEGIN                                                      04535
      IF auxinflag % reading from aux. input source %            04536
         THEN auxinterminate();                                  04537
      typech( 7B ); % ? %                                        04538
      END;                                                       04539
   =echostr:        % echo supplied string in cfl %             04540
      IF fbackmode = verbsmode THEN                              04541
         BEGIN                                                   04542
         *tempstr* ← '(,                                         04543
            *[p1]*[1 TO MIN([p1].L, feedbk)], ');                04544
         feedbk ← feedbk+2;                                      04545
```

```
        ON SIGNAL ELSE feedbk ← feedbk-2;                    04546
        echo( $tempstr );                                    04547
        feedbk ← feedbk-2;                                   04548
        ON SIGNAL ELSE;                                      04549
        todco( SP );                                         04550
        END;                                                 04551
=rechostr:      % replace previously supplied string in
cfl %                                                        04552
    BEGIN                                                    04553
    *tempstr* ← '(, *[pl]*/1 TO MIN([pl].L, feedbk)], ');
                                                             04554
    echo( $"← " );                                           04555
    feedbk ← feedbk+2;                                       04556
    ON SIGNAL ELSE feedbk ← feedbk-2;                        04557
    echo( $tempstr );                                        04558
    feedbk ← feedbk-2;                                       04559
    ON SIGNAL ELSE;                                          04560
    END;                                                     04561
=typelit, =typecalit:   % put up literal string (not in
cfl) %                                                       04562
    BEGIN                                                    04563
    crlf();                                                  04564
    typeas( pl );                                            04565
    END;                                                     04566
=typenullit:   % put up null lit string (not in cfl) %
                                                             04567
    BEGIN                                                    04568
    crlf();                                                  04569
    END;                                                     04570
=addcalit:      % put up literal string (not in cfl) %
                                                             04571
    NULL;                                                    04572
    %---- don't need to do anything if device is
    typewriter ------%                                       04573
=startparams:   % begin parameter collection %               04574
    BEGIN                                                    04575
    echoff();                                                04576
    END;                                                     04577
=incues:        % provide inpt prompts %                    04578
    IF inprompt # noprompts THEN                             04579
        IF NOT cueflg THEN                                   04580
            BEGIN                                            04581
            cueflg ← TRUE;                                   04582
            typeas( pl );                                    04583
            typech( SP );                                    04584
            END;                                             04585
=fbpop:                 % cleanup after popping states %
                                                             04586
    BEGIN                                                    04587
    cueflg ← FALSE;                                          04588
    echo( $" ← ");                                           04589
    END;                                                     04590
=fbaddlit, =fbendlit:   % add pl to literal area %           04591
    BEGIN                                                    04592
    typeas( pl );                                            04593
    END;                                                     04594
```

```
            ENDCASE err($"Illegal call to fbctl");                    04595
         END;                                                         04596
      RETURN;                                                         04597
      END.
                                                                      04598
% HELP ROUTINES %                                                    02566
   (fbhelp) PROCEDURE( list, cmode ); % lists alternatives %          06337
      LOCAL                                                           06338
         svllcmd, svrec1, svrec2, ncols, hlpblk, achn, cinstptr,
                                                                      06339
         savecnptr,                                                   06340
         lookadr, % address of name of string to be found in help DB
         if in help %                                                 06341
         nextchar,  % next inpt character %                           06342
         hindex,    % hit index into vocab for keyword %             06343
         helpcnt,   % number of keywords in helpstr %                 06344
         % work SHOULD BE wasize (in FDATA) WORDS LONG %              06345
            work[30];        % state buffer for keyinit and nextkey %
                                                                      06346
      LOCAL TEXT POINTER ptr, t1;                                     06347
      LOCAL STRING                                                    06348
         lev2str[10],        % level 2 command promtp %              06349
         keywordstr[25],     % contains keywordstr %                 06350
         helpstr[2000];      % contains subhelp info %               06351
      % REF VARIABLES %                                              06352
         REF fbstr, inpt, cinstptr;                                   06353
         REF tda, litda, sysmsg, hlpcmdstk;                           06354
         REF list;                                                    06355
      %---------------------%                                        06356
      % Do special things in help. %                                 06357
         IF inhelp THEN                                               06358
            BEGIN                                                     06359
            *helpstr* ← "The question mark message for Help is
            missing.  Call ARC!";                                     06360
            IF hlpfileno THEN                                         06361
               BEGIN                                                  06362
               lookadr ←                                              06363
                  IF nlmode = fulldisplay THEN $"dqmark" ELSE
                  $"tqmark";                                          06364
               ptr ← orgstid;                                         06365
               ptr.stfile ← hlpfileno;                                06366
               ptr[1] ← 1;                                            06367
               lookup($ptr, lookadr, nametyp);                        06368
               IF ptr # endfil THEN                                   06369
                  BEGIN                                               06370
                  % Remove name, first EOL. %                         06371
                  IF FIND SF(ptr) [EOL] ↑t1 THEN                      06372
                     *helpstr* ← t1 SE(ptr);                          06373
                  END;                                                06374
               END;                                                   06375
            fbctl( typecalit, $helpstr);                              06376
            RETURN;                                                   06377
            END;                                                      06378
      % initialize %                                                 06379
         *lev2str* ← "<>";                                            06380
         ncols ← (litda.daright - litda.daleft) / litda.dahinc;       06381
```

```
      ncols ← IF nlmode = fulldisplay THEN ncols/3 ELSE
      tda.damcol/3;                                                    06382
   svrec1 ← recogmode;                                                 06383
   svrec2 ← recog2mode;                                                06384
   svllcmd ← FALSE;                                                    06385
   hlpblk ← achn ← 0;                                                  06386
% save current node pointer in path stack %                           06387
   savecnptr ← list.curnodeptr;                                        06388
% cleanup on any errors %                                             06389
   ON SIGNAL ELSE                                                      06390
      BEGIN                                                            06391
      ON SIGNAL ELSE;                                                  06392
      IF hlpblk THEN dlfmb( hlpblk := 0);                              06393
      IF svllcmd AND svrec1 = mexpert THEN                             06394
         BEGIN                                                         06395
         recogmode ← svrec1;                                           06396
         recog2mode ← svrec2;                                          06397
         END;                                                          06398
      savecnptr ← list.curnodeptr;                                     06399
      END;                                                             06400
% diddle around to get all possibilites %                             06401
   svllcmd ← cmode.llcmd := FALSE;                                     06402
   IF svllcmd AND recogmode = mexpert THEN                             06403
      recogmode ← recog2mode ← manticipatory;                          06404
% get storage for building strings %                                  06405
   IF NOT hlpblk ← dlgmb( 400 ) THEN GOTO fbrhelp;                     06406
% set up some state info (change later) %                             06407
   % *** note:  we are using a new work area to control the
   sequencing through the keyword list, but we still use the
   fbstr from the work area presently in effect %                      06408
      *helpstr* ← "Current Alternatives are:
      ";                                                               06409
      helpcnt ← 0; %number of words in helpstr %                       06410
      nextchar ← -1; % so nextkey will give back next thing %
                                                                       06411
      keyinit( &list, $work, cmode, TRUE);                             06412
      &cinstptr ← work.currinst;                                       06413
% use nextkey to get all keywords on current list and add them
to the helpstring %                                                    06414
   WHILE (hindex ← nextkey( $work, nextchar )) # 0 DO                  06415
      BEGIN                                                            06416
      IF inptrf THEN EXIT LOOP;                                        06417
      IF NOT                                                           06418
         addhelp( hindex.RH, hlpblk, $work, ncols, $lev2str,
         svllcmd, $achn, $helpcnt, svrec1, &cinstptr) THEN
         EXIT LOOP;                                                    06419
      BUMP helpcnt;                                                    06420
      &cinstptr ← work.currinst;                                       06421
      END;                                                             06422
% output the helpstring %                                             06423
   fbbldhlp( $helpstr, achn, helpcnt, ncols);                          06424
(fbrhelp):                                                             06425
% restore current node pointer in pathst %                           06426
   list.curnodeptr ← savecnptr;                                        06427
IF hlpblk THEN dlfmb( hlpblk := 0);                                    06428
IF svllcmd AND svrec1 = mexpert THEN                                   06429
```

```
        BEGIN                                                   06430
        recogmode ← svrec1;                                     06431
        recog2mode ← svrec2;                                    06432
        END;                                                    06433
    RETURN;                                                     06434
    END.
                                                                06435
(addhelp) PROCEDURE( index, hlpblk, work, ncols, lev2str, clv1,
acnn, helpcnt, reclmd, cinstptr ); % edits name and adds to
helpstr %                                                       04648
    LOCAL                                                       04649
        ablk, temp, pscnt,                                      04650
        nsptr,      % ptr for getting noise words %              04651
        char,       % first char in keywordstr %                04652
        i,                                                      04653
        l;                                                      04654
    LOCAL TEXT POINTER tp1, tp2, tp3;                           04886
    LOCAL STRING                                                04655
        blnkstr[10],                                            04656
        nsstr[100],                                             04657
        keywordstr[100];    % vocabulary string %              04658
    REF helpcnt, achn, ablk, lev2str, work, nsptr, cinstptr;    04659
    %--------------------%                                      04660
    % initialize balnk string %                                04891
        *blnkstr* ← "          ";                               04661
    % deal with INSERT and REPEAT Commands and ↑Q %             04892
        IF NOT helpcnt THEN                                     05923
            BEGIN                                               05924
            IF NOT add2help($"<CTRL-Q> for HELP", hlpblk, &lev2str,
            &achn, ncols, clv1) THEN RETURN( FALSE );           05926
            BUMP helpcnt;                                       05927
            IF NOT add2help($"<CTRL-S> for SYNTAX", hlpblk,
            &lev2str, &achn, ncols, clv1) THEN RETURN( FALSE ); 07110
            BUMP helpcnt;                                       07111
            IF basestateflag AND (NOT kwrstate) AND (clv1 OR reclmd
            # mexpert) THEN                                     04662
                BEGIN                                           04663
                IF  [$sbstack + sbstkx - $sbentsize].sbptr =
                $nlseditor THEN                                 04888
                    BEGIN                                       04889
                    IF NOT add2help($"OKINSERT", hlpblk, &lev2str,
                    &achn, ncols, clv1) THEN RETURN( FALSE );   04664
                    BUMP helpcnt;                               04665
                    END;                                        04890
                IF NOT add2help($"OKREPEAT", hlpblk, &lev2str, &achn,
                ncols, clv1) THEN RETURN( FALSE );              04666
                BUMP helpcnt;                                   04667
                END;                                            04668
            END;                                                05925
    % take care of parse function prompts %                     04878
        IF cinstptr.opcode = $pfcall THEN                       04881
            BEGIN                                               04882
            *keywordstr* ← NULL;                                04910
            % call the parsefunction to get string %            04915
              [cinstptr.addr]( 0, parseqmark, $keywordstr);     04893
            % find and handle first entity %                    04916
```

```
            *nsstr* ← 0;                                                    04905
            FIND SF(*k#ywordstr*) ↑tp1                                      04894
               [ *nsstr* ↑tp3 ↑tp2 ←tp2 / ENDCHR ↑tp3 ↑tp2];                04906
            *nsstr* ← tp1 tp2;                                              04896
            IF nsstr.L THEN                                                 04911
               BEGIN                                                        04917
               IF NOT add2help($nsstr, h1pb1k, &lev2str, &achn,
               ncols, c1v1) THEN RETURN( FALSE );                           04895
               END                                                          04918
            ELSE RETURN( TRUE );                                            04919
         % deal with the remaining (if any) entities %                      04920
            LOOP                                                            04897
               BEGIN                                                        04898
               *nsstr* ← 0;                                                 04907
               FIND tp3 > ↑tp1                                              04899
                  [ *nsstr* ↑tp3 ↑tp2 ←tp2 / ENDCHR ↑tp3 ↑tp2];
                                                                            04921
               *nsstr* ← tp1 tp2;                                           04900
               IF nsstr.L THEN                                              04922
                  BEGIN                                                     04924
                  IF NOT add2help($nsstr, h1pb1k, &lev2str,
                  &achn, ncols, c1v1) THEN RETURN( FALSE );                 04901
                  END                                                       04925
               ELSE EXIT LOOP;                                              04923
               BUMP helpcnt;                                                04902
               END;                                                         04904
            RETURN( TRUE );                                                 04903
            END;                                                            04885
      % get a block for this entry %                                        04669
         IF NOT &ab1k ←                                                     04670
            d1gtb1k( h1pb1k, ((ncols+4)/5)+1+$fbh1p1 ) THEN RETURN(
            FALSE );                                                        04671
         ab1k.nbastr ← &ab1k+$fbh1p1;                                       04672
         [ab1k.nbastr].M ← ncols;                                           04673
      % get symbolic represenation for the keyword %                       04674
         [$keywordstr+1] ← 0;                                               04675
         vocab( index, $keywordstr );                                       04676
      % get value of this keyword for sorting %                            04677
         ab1k.nbval ← [$keywordstr+1] / 2;                                  04678
      % get print representation for non-printing symbols (if any) %
                                                                            04679
         IF keywordstr.L = 1                                                04680
            AND (char ← *keywordstr*[1]) < 40B % non printing chars
            %                                                               04681
               THEN *keywordstr* ← *[ npstrad( char ) ]*;                   04682
      % test for optional keywords %                                       04683
         IF work.optcount AND NOT work.optflag                              04684
            THEN *keywordstr* ← '[, *keywordstr*, '];                       04685
      % take care of level 2 commands %                                    04686
         IF c1v1 THEN                                                       04687
            BEGIN                                                           04688
            &nsptr ← work.currpath;                                         04689
            &nsptr ← nsptr.curnodeptr;                                      04690
            IF nsptr.opcode = $keyop AND NOT nsptr.ctr1.11cmd THEN
                                                                            04691
               *keywordstr* ← *lev2str*, *keywordstr*                       04692
```

```
        ELSE                                                    04693
            *keywordstr* ← *blnkstr*[1 TO lev2str.L],
            *keywordstr*;                                       04694
        END;                                                    04697
    % edit in noise words %                                     04696
     &nsptr ← work.currpath;                                    04697
     &nsptr ← nsptr.curnodeptr;                                 04698
     *nsstr* ← NULL;                                            04699
     WHILE &nsptr ← nsptr.nsuccessor DO                         04700
        IF nsptr.opcode # $necno THEN EXIT LOOP                 04701
        ELSE                                                    04702
            IF nsstr.L THEN                                     04703
               *nsstr* ← *nsstr*, SP, *[nsptr.addr]*            04704
            ELSE                                                04705
               *nsstr* ← *[nsptr.addr]*;                        04706
    % edit the string %                                         04707
     temp ← ncols - keywordstr.L - 4;                           04708
     IF nsstr.L > temp THEN                                     04709
        *nsstr* ← *nsstr*[1 TO (temp-3)], "...";                04710
     IF nsstr.L THEN                                            04711
        *[ablk.hbastr]* ← *keywordstr*, " (", *nsstr*, ')'      04712
     ELSE                                                       04713
        *[ablk.hbastr]* ← *keywordstr*;                         04714
    % put this block where it belongs %                         04715
     IF NOT (&nsptr ← achn.hbfor) THEN achn.hbfor ← &ablk       04716
     ELSE                                                       04717
        IF ablk.hbval < nsptr.hbval THEN % special for first one
        %                                                       04718
            BEGIN                                               04719
            ablk.hbfor ← &nsptr;                                04720
            achn.hbfor ← nsptr.hbbck ← &ablk;                   04721
            END                                                 04722
        ELSE                                                    04723
            CASE TRUE OF                                        04724
               ENDCASE                                          04725
                  IF NOT nsptr.hbfor THEN                       04726
                     BEGIN                                      04727
                     nsptr.hbfor ← &ablk;                       04728
                     ablk.hbbck ← &nsptr;                       04729
                     END                                        04730
                  ELSE                                          04731
                     IF ablk.hbval < [nsptr.hbfor].hbval THEN
                                                                04732
                        BEGIN                                   04733
                        nsptr.hbfor.hbbck ← &ablk;              04734
                        ablk.hbfor ← nsptr.hbfor;               04735
                        ablk.hbbck ← &nsptr;                    04736
                        nsptr.hbfor ← &ablk;                    04737
                        END                                     04738
                     ELSE                                       04739
                        BEGIN                                   04740
                        &nsptr ← nsptr.hbfor;                   04741
                        REPEAT CASE;                            04742
                        END;                                    04743
    RETURN( TRUE );                                             04744
    END.
```

```
                                                                    04745
(fbbldhlp) PROCEDURE ( helpstr, achn, helpcnt, ncols);              04599
   LOCAL nrows, i, j, temp, tptr;                                   04600
   LOCAL STRING blnkstr[40];                                        04601
   REF helpstr, achn, tptr;                                         04602
   nrows ← (helpcnt+2)/3;                                           04603
   *blnkstr* ← "                                    ";              04604
   fbctl( typenulllit, );                                           04605
   fbctl( fbaddlit, &helpstr);                                      04606
   FOR i ← 0 UP UNTIL = nrows DO                                    04607
      IF &achn THEN                                                 04608
         BEGIN                                                      04609
         IF (inptrf := FALSE) THEN EXIT LOOP;                       04610
         *helpstr* ← NULL;                                          04611
         % do first one %                                           04612
            temp ← ncols - [achn.hbastr].L;                         04613
            *helpstr* ←                                             04614
               *helpstr*, *[achn.hbastr]*, *blnkstr*[1 TO temp];
                                                                    04615
         % do second one %                                          04616
            &tptr ← &achn;                                          04617
            FOR j ← 1 UP UNTIL > nrows DO                           04618
               IF &tptr THEN &tptr ← tptr.hbfor                     04619
               ELSE EXIT LOOP;                                      04620
            IF &tptr THEN                                           04621
               BEGIN                                                04622
               temp ← ncols - [tptr.hbastr].L;                      04623
               *helpstr* ←                                          04624
                  *helpstr*, *[tptr.hbastr]*, *blnkstr*[1 TO
                  temp];                                            04625
               END;                                                 04626
         % do third one %                                           04627
            IF &tptr THEN                                           04628
               BEGIN                                                04629
               FOR j ← 1 UP UNTIL > nrows DO                        04630
                  IF &tptr THEN &tptr ← tptr.hbfor                  04631
                  ELSE EXIT LOOP;                                   04632
               IF &tptr THEN                                        04633
                  *helpstr* ← *helpstr*, *[tptr.hbastr]*;           04634
               END;                                                 04635
         % finish the line %                                        04636
            *helpstr* ← *helpstr*, EOL;                             04637
         % give the user the line %                                 04638
            fbctl( fbaddlit, &helpstr );                            04639
         % go to the next line %                                    04640
            &achn ← achn.hbfor;                                     04641
         END                                                        04642
      ELSE EXIT LOOP;                                               04643
   *helpstr* ← EOL, "---", *fbstr*;                                 04644
   fbctl( fbendlit, &helpstr);                                      04645
   RETURN;                                                          04646
   END.
                                                                    04647
(add2help) PROCEDURE( kstring, hlpblk, lev2str, achn, ncols,
clvl ); % edits name and adds to helpstr %                          04819
   LOCAL                                                            04820
```

```
    ablk, temp,                                                    04821
    nsptr,      % ptr for getting noise words %                     04822
    char,       % first char in keywordstr %                        04823
    i,                                                             04824
    l;                                                             04825
LOCAL STRING                                                       04826
    blnkstr[10],                                                   04827
    nsstr[100],                                                    04828
    keywordstr[100];    % vocabulary string %                       04829
REF kstring, achn, ablk, lev2str, nsptr;                           04830
%--------------------%                                            04831
*blnkstr* ← "          ";                                          04832
% get a block for this entry %                                     04833
    IF NOT &ablk ←                                                 04834
       dlgtblk( hlpblk, ((ncols+4)/5)+1+$fbhlpl ) THEN RETURN(
       FALSE );                                                    04835
    ablk.hbastr ← &ablk+$fbhlpl;                                   04836
    [ablk.hbastr].M ← ncols;                                       04837
% get symbolic represenation for the keyword %                     04838
    *keywordstr* ← *kstring*;                                      04839
% get value of this keyword for sorting %                          04840
    ablk.hbval ← 35M;                                              04841
% take care of level 2 commands %                                  04842
    IF clvl THEN                                                   04843
       *keywordstr* ← *blnkstr*[1 TO lev2str.L], *keywordstr*;
                                                                   04844
% edit the string %                                               04845
    *[ablk.hbastr]* ← *keywordstr*;                                04846
% put this block where it belongs %                               04847
    IF NOT (&nsptr ← achn.hbfor) THEN achn.hbfor ← &ablk           04848
    ELSE                                                           04849
       IF ablk.hbval < nsptr.hbval THEN % special for first one
       %                                                           04850
          BEGIN                                                    04851
          ablk.hbfor ← &nsptr;                                     04852
          achn.hbfor ← nsptr.hbbck ← &ablk;                        04853
          END                                                      04854
       ELSE                                                        04855
          CASE TRUE OF                                             04856
             ENDCASE                                               04857
                IF NOT nsptr.hbfor THEN                            04858
                   BEGIN                                           04859
                   nsptr.hbfor ← &ablk;                            04860
                   ablk.hbbck ← &nsptr;                            04861
                   END                                             04862
                ELSE                                               04863
                   IF ablk.hbval < [nsptr.hbfor].hbval THEN
                                                                   04864
                      BEGIN                                        04865
                      nsptr.hbfor.hbbck ← &ablk;                   04866
                      ablk.hbfor ← nsptr.hbfor;                    04867
                      ablk.hbbck ← &nsptr;                         04868
                      nsptr.hbfor ← &ablk;                         04869
                      END                                          04870
                   ELSE                                            04871
                      BEGIN                                        04872
```

```
                        &nsptr ← nsptr.hbfor;                          04873
                        REPEAT CASE;                                   04874
                        END;                                           04875
    RETURN( TRUE );                                                    04876
    END.
                                                                       04877
% <↑S> syntax help routines %                                         06916
  (cshelp) PROCEDURE( list, cmode, fstrm ); % lists alternatives %
                                                                       06917
    LOCAL                                                              06918
      svllcmd, svrec1, svrec2, cinstptr,                               06919
      savecnptr, lookadr,                                              06920
      nextchar,  % next inpt character %                               06921
      hindex,    % hit index into vocab for keyword %                   06922
      % work SHOULD BE wasize (in PDATA) WORDS LONG %                  06923
        work[30];       % state buffer for keyinit and nextkey %
                                                                       06924
    LOCAL TEXT POINTER ptr, t1;                                        06925
    LOCAL STRING helpstr[300];                                         07135
    % REF VARIABLES %                                                 06926
      REF fbstr, inpt, cinstptr;                                       06927
      REF tda, litda, sysmsg, hlpcmdstk;                               06928
      REF list;                                                        06929
    %----------------------%                                          06930
    % Do special things in help. %                                    07113
      IF inhelp THEN                                                   07114
        BEGIN                                                          07115
        *helpstr* ← "The question mark message for Help is
        missing.  Call ARC!";                                          07116
        IF hlpfileno THEN                                              07117
          BEGIN                                                        07118
          lookadr ←                                                    07119
            IF nlmode = fulldisplay THEN $"dqmark" ELSE
            $"tqmark";                                                 07120
          ptr ← orgstid;                                               07121
          ptr.stfile ← hlpfileno;                                      07122
          ptr[1] ← 1;                                                  07123
          lookup($ptr, lookadr, nametyp);                              07124
          IF ptr # endfil THEN                                         07125
            BEGIN                                                      07126
            % Remove name, first EOL. %                                07127
            IF FIND SF(ptr) [EOL ) !T1 THEN                            07128
              *helpstr* ← t1 SE(ptr);                                  07129
            END;                                                       07130
          END;                                                         07131
        fbctl( typecalit, $helpstr);                                   07132
        RETURN;                                                        07133
        END;                                                           07134
    % initialize %                                                    06937
      svrec1 ← recogmode;                                              06938
      svrec2 ← recog2mode;                                             06939
      svllcmd ← FALSE;                                                 06940
      IF nlmode = fulldisplay THEN cmdctl ← 1                          06941
      ELSE cmdctl ← 2;                                                 06942
    % save current node pointer in path stack %                       06943
      savecnptr ← list.curnodeptr;                                     06944
```

```
% cleanup on any errors %                                        06945
   ON SIGNAL ELSE                                                06946
      BEGIN                                                      06947
      ON SIGNAL ELSE;                                            06948
      IF svllcmd AND svrecl = mexpert THEN                       06949
         BEGIN                                                   06950
         recogmode ← svrecl;                                     06951
         recog2mode ← svrec2;                                    06952
         END;                                                    06953
      savecnptr ← list.curnodeptr;                               06954
      END;                                                       06955
% set up output display area %                                  06956
   fbctl( typenulllit );                                         06957
% build a stack of nodes from the pathh stack %                 06958
   CASE prspathstk( cmode ) OF                                   06959
      = 0:                                                       06960
         BEGIN                                                   06961
         &cinstptr ← [pathstk.begnodeptr].addr;                  06962
         WHILE &cinstptr DO                                      06963
            BEGIN                                                06964
            ckshwcmd( &cinstptr );                               06965
            &cinstptr ← cinstptr.alternative;                    06966
            END;                                                 06967
         &cinstptr ← [[pathstk.begnodeptr].alternative].addr;    06968
         WHILE &cinstptr DO                                      06969
            BEGIN                                                06970
            ckshwcmd( &cinstptr );                               06971
            &cinstptr ← cinstptr.alternative;                    06972
            END;                                                 06973
         END;                                                    06974
      = 1: ckshwcmd( csstk[0] );                                 06975
      ENDCASE                                                    06976
         BEGIN                                                   06977
         % diddle around to get all possibilites %               06978
            svllcmd ← cmode.llcmd := FALSE;                      06979
            IF svllcmd AND recogmode = mexpert THEN              06980
               recogmode ← recog2mode ← manticipatory;           06981
         % set up some state info (change later) %              06982
            % *** note:  we are using a new work area to
            control the sequencing through the keyword list,
            but we still use the fbstr from the work area
            presently in effect %                                06983
               nextchar ← -1; % so nextkey will give back next
               thing %                                           06984
               keyinit( &list, $work, cmode, TRUE);              06985
         % loop with nextkey to get all paths %                 06986
            LOOP                                                 06987
               BEGIN                                             06988
               IF NOT nextkey( $work, nextchar) THEN EXIT
               LOOP;                                             06989
               csstk[csstkx - 1] ← [work.currpath].curnodeptr;
                                                                 06990
               csstkb ← 0;                                       06991
               ckshwcmd( csstk[0] );                             06992
               END;                                              06993
```

```
            IF svllcmd AND svrec1 = mexpert THEN                    06994
               BEGIN                                                06995
               recogmode ← svrec1;                                  06996
               recog2mode ← svrec2;                                 06997
               END;                                                 06998
            END;                                                    06999
      % output the helpstring %                                     07000
         inptrf ← 0;                                                07112
         IF nlmode = fulldisplay THEN fbctl( addcalit )             07001
         ELSE                                                       07144
            BEGIN                                                   07145
            *helpstr* ← EOL, "---";                                 07147
            IF fstrm THEN *helpstr* ← *helpstr*, *fbstr*;           07149
            fbctl( fbendlit, $helpstr);                             07148
            END;                                                    07146
      % restore current node pointer in pathst %                   07002
         list.curnodeptr ← savecnptr;                               07003
      RETURN;                                                       07004
      END.
                                                                    07005
   (prspathstk) PROCEDURE                                           07006
      % FORMALS %                                                   07007
         ( cmode );          % current recognition mode %           07008
      % LOCALS %                                                    07009
         LOCAL j, jtop, pathptr;                                    07010
      REF pathptr;                                                  07011
                                                                    07012
      csstkx ← csstkb ← 0;                                          07013
      jtop ← pathx / $totalrecsize;                                 07014
      FOR j ← 0 UP UNTIL = jtop DO                                  07015
         BEGIN                                                      07016
         &pathptr ← $pathstk + (j*$totalrecsize);                   07017
         CASE j OF                                                  07018
            = 0:                                                    07019
               csstk[csstkx := csstkx+1] ← pathptr.curnodeptr;      07020
            ENDCASE                                                 07021
               CASE pathptr.begnodeptr OF                           07022
                  = pathptr.curnodeptr:                             07023
                     csstk[csstkx := csstkx+1] ← pathptr.curnodeptr;
                                                                    07024
                  ENDCASE fndwhere( &pathptr );                     07025
         END;                                                       07026
      CASE csstkx OF                                                07027
         = 1:                                                       07028
            IF pathstk.begnodeptr = pathstk.curnodeptr THEN         07029
               BEGIN                                                07030
               csstkx ← -1;                                         07031
               RETURN( 0 );                                         07032
               END                                                  07033
            ELSE REPEAT CASE( 2 );                                  07034
         ENDCASE                                                    07035
            CASE [csstk[csstkx-1]].opcode OF                        07036
               = $keyop:                                            07037
                  CASE kwrstate OF                                  07038
                     < 0:  RETURN( 1 );                             07039
                     = 0:                                           07040
```

```
                        BEGIN                                                07041
                        BUMP DOWN csstkx;                                    07042
                        RETURN( 1 );                                         07043
                        END;                                                 07044
                     > 0:  RETURN( 2 );                                      07045
                     ENDCASE;                                                07046
               ENDCASE                                                       07047
                  BEGIN                                                      07048
                  CASE /csstk/csstkx-1]].opcode OF                           07049
                     = $option, = $pfcall: BUMP DOWN csstkx;                 07050
                     ENDCASE;                                                07051
                  RETURN( 1 );                                               07052
                  END;                                                       07053
END.
                                                                             07054
%%                                                                           07055
```

```
(fndwhere) PROCEDURE                                                    07056
   % FORMALS %                                                          07057
      (pathptr);                                                        07058
   % LOCALS %                                                          07059
      LOCAL instptr;                                                    07060
   REF pathptr, instptr;                                                07061
                                                                        07062
   &instptr ← pathptr.begnodeptr;                                       07063
   WHILE &instptr DO                                                    07064
      IF &instptr = pathptr.curnodeptr THEN                             07065
         BEGIN                                                          07066
         csstk/csstkx := csstkx+1 ] ← pathptr.curnodeptr;               07067
         RETURN;                                                        07068
         END                                                            07069
      ELSE CASE instptr.opcode OF                                       07070
         = $xecute, = $option:                                          07071
            IF fndxec( &instptr, pathptr.curnodeptr ) THEN RETURN
                                                                        07072
            ELSE REPEAT CASE( -1 );                                     07073
         ENDCASE                                                        07074
            IF &instptr = instptr.alternative THEN &instptr ← 0
                                                                        07075
            ELSE &instptr ← instptr.alternative;                        07076
   RETURN;                                                              07077
   END.
                                                                        07078
   %%                                                                  07079
```

```
(fndxec) PROCEDURE                                                      07080
   % FORMALS %                                                          07081
      (xecptr,                                                          07082
      objptr);                                                          07083
   % LOCALS %                                                          07084
      LOCAL instptr;                                                    07085
   REF instptr, xecptr;                                                 07086
                                                                        07087
   &instptr ← xecptr.addr;                                              07088
   WHILE &instptr DO                                                    07089
      IF &instptr = objptr THEN                                         07090
         BEGIN                                                          07091
         csstk[csstkx := csstkx+1] ← &xecptr;                           07092
         csstk[csstkx := csstkx+1] ← objptr;                            07093
         RETURN( TRUE );                                                07094
         END                                                            07095
      ELSE CASE instptr.opcode OF                                       07096
         = &xecute, = $option:                                          07097
            BEGIN                                                       07098
            csstk[csstkx := csstkx+1] ← &xecptr;                        07099
            IF fndxec( &instptr, objptr) THEN RETURN( TRUE )            07100
            ELSE BUMP DOWN csstkx;                                      07101
            REPEAT CASE( -1 );                                          07102
            END;                                                        07103
         ENDCASE                                                        07104
            IF &instptr = instptr.alternative THEN &instptr ← 0
                                                                        07105
            ELSE &instptr ← instptr.alternative;                        07106
   RETURN( FALSE );                                                     07107
   END.
                                                                        07108
   %%                                                                  07109
```

FINISH 02625

PDATA

(MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PD
(MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PD

```
< NLS, PDATA.NLS;13, >, 25-OCT-74 09:52 KJM ;;;;
FILE pdata  %  L10  <REL-NLS>PDATA  %% (L10,) (rel-nls,PDATA.rel,) %
                                                                    02

% CONSTANTS %                                                      0154
   % SIZE DECLARATIONS %                                           0155
      SET EXTERNAL                                                  0156
         prsvsize = 30,          % room for 10 parse rule replacements %
                                                                    0212
         funrecsize = 10,        % size of funstaterec %           0157
         pathrecsize = 3,        % size of pathsr %                0158
         totalrecsize = 13,      % total rec size %                0159
         psdepth = 200,          % total  entries in path stack%   0160
         pssize = 2600,          % psdepth * totalrecsize %        0161
         selstatesize = 3,       % size of selstate record %       0162
         ptrssize = 20,          % size of ptrstk %                0163
            % CHANGE DECLARATIONS IN PARSER ALSO %                 0213
               % use following pattern to find spots to change    0215
                 /"ptrssize (in PDATA)"/;   %                       0216
         evalsize = 40,          % size of eval stack %            0164
         sbstksize = 20,         % size of sbstack %               0165
         sbentsize = 2,          % size of sbentry record %        0166
         framesize = 4,          % size of subsystem definition frame %
                                                                    0167
         sdptsize = 84,          % size of nlssubs and allsubs %   0168
         fbstrmax = 50,          % max size of feedback string %   0169
         wasize = 15;            % work area word size (5+fbstrmax/5) %
                                                                    0170
            % CHANGE DECLARATIONS IN PARSER ALSO %                 0218
               % use following pattern to find spots to change    0219
                 /"wasize (in PDATA)"/;   %                         0220
   % VALIDATION CODE FOR SUBSYSTEM DISPATCH RECORD %               0172
      % This 18 bit validation code is output by the CML compiler and
      checked by dfnsubsys routine to make sure that the pointer being
      handed that routine actually points to a subsystem dispatch
      record %                                                      0173
      SET EXTERNAL dptvldationcode = 110473;                        0174
   % KEYWORD DELIMITER CHARS %                                     0175
      SET EXTERNAL                                                  0176
         eschar = 1B0B,          % ESC for recognition purposes %  0177
         leftdelim =SP;          % left delimiter for keywords %   0178
   % INTERPRETER EVALUATION MODES %                                0179
      SET EXTERNAL                                                  0180
         unknown = 0,            % dont know what type it is %     0181
         parallel = 1,           % parallel recognition %          0182
         serial = 2,             % serial recognition %            0183
         parsefunction = 3;      % interpretted by parsing function % 0184
   % INTERPRETER OPCODE DECLARATIONS %                             0185
      SET EXTERNAL                                                  0186
         % RECOGNIZERS %                                          0187
            keyop=1B,            % keyword recognition operator %  0188
            confirm=2B,          % get command confirmation %      0189
            ssel=3B,             % source selection %              0190
            dsel=4B,             % destination selection %         0191
            lsel=5B,             % literal selection %             0192
            pca=6B,              % ca recognizer %                 0193
            vwspecs=7B,          % gets viewspecs %                0194
```

```
        levadj=10B,          % get level adjust string %                  0195
      % OPTIONAL ELEMENTS %                                               0196
        option=11B,          % optional parameter %                       0197
        anyof=12B,           % repeated list with failure %               0198
      % CONTROL ELEMENTS %                                                0199
        pfcall=21B,          % parse function call %                      0200
        xecute=22B,          % transfer to another point in tree %
                                                                          0201
        call=23B,            % subroutine call %                          0202
      % FEEDBACK ELEMENTS %                                               0203
        fbclear=31B,         % clear feedback buffer %·                   0204
        necho=32B,           % echo noise word string %                   0205
        recho=33B,           % replace last thing echoed %                0206
      % VALUE MANIPULATIONS %                                             0207
        store=41B,           % store value into variable %                0208
        pload=42B,                 % load variable to ptr stack %
                                                                          0209
        enter=43B,           % enter constant into stack %                0210
        valueof=44B;         % valueof built in function %                0211
% parser flags %                                                           03
  DECLARE EXTERNAL                                                        04
  littakedown = 1;     % TRUE IFF literal is to be taken down (see
  <select,getlit> %                                                       05
% repeat command state info %                                            06
  DECLARE EXTERNAL STRING                                                 07
    keyinpstr[50];   % actual chars typed to get recognition %           08
  DECLARE EXTERNAL STRING                                                 09
    keyrptstr[50];   % recognition string at end of last command %       010
  DECLARE EXTERNAL                                                        011
    keysaveflag = 1;         % TRUE if keyword inpt chars are to be
    saved %                                                               012
% parser variables %                                                      013
  DECLARE EXTERNAL                                                        014
    %directory and archive directory commands%                            015
      gparam,                                                             016
      gparm2,                                                             017
      gparm3,                                                             018
      gparm4,                                                             019
    aheadfilename,                                                        020
    dent,                                                                 021
    dest,                                                                 022
    ent,                                                                  023
    filtre,                                                               024
    namfil,                                                               025
    ff,                                                                   026
    fromwhom,                                                             027
    level,                                                                028
    literal,                                                              029
    param,                                                                030
    param1,                                                               031
    param2,                                                               032
    param3,                                                               033
    param4,                                                              0149
    pb,                                                                   034
    port,                                                                 035
    retfilename,                                                          036
```

```
    s1,                                                                 037
    s2,                                                                 038
    sent,                                                               039
    sim,                                                                040
    source,                                                             041
    vs;                                                                 043
% writeable data for parser %                                          044
  % INTERPRETER CONTROL VARIABLES %                                     045
    % CURRENT MARKER %                                                 054
      DECLARE EXTERNAL TEXT POINTER                                     055
        curmkr;    % current marker %                                   056
      DECLARE EXTERNAL                                                  057
        cspupdate,          %flag to set up csp from curmkr in dnls
        -- contains da address or zero%                                058
        usesrr,    %address of statement return ring to use for
        jump file return or zero%                                       059
        cspcacode, %content analyzer code address -- see CMDFINISH%
                                                                        060
        cspusqcod, %user sequence generator program address -- see
        CMDFINISH%                                                      061
        cspvs[2];  %viewspecs to be used with curmkr if changecsp
        is set%                                                         062
    % PTR TO THE INPUT ROUTINE %                                       063
      DECLARE EXTERNAL                                                  064
        inpt;                                                           065
    % CURRENT GRAMMAR %                                                066
      DECLARE EXTERNAL                                                  067
        grammar;   % ptr to start of grammar %                         068
    % FRAME COUNTER %                                                  069
      DECLARE EXTERNAL                                                  070
        framecounter = 0;                                               071
    % CUT BACK POINTER %                                               0224
      DECLARE EXTERNAL                                                  0225
        cutstop = 0;                                                    0226
    % TERMINATION CHARACTERS FOR NLS COMMANDS %                        072
      DECLARE EXTERNAL                                                  073
        cachar=4,           % ↑D %                                      074
        optchar=25B,        % ↑U %                                      075
        rptchar=140B,       % ↑B %                                      076
        inschar=5;                  % ↑E %                            077
  % INTERPTETER STATE DATA %                                           078
    % PROMPTS STRING %                                                 079
      DECLARE EXTERNAL STRING                                           080
        prompstr[25];                                                   081
    % COMMAND HEARALD STRING %                                         082
      DECLARE EXTERNAL STRING hrldstr[30];  % TNLS hearald string %
                                                                        083
    % STRING FOR PROMPTS FOR PFCALL FUNCTIONS %                       084
      DECLARE EXTERNAL STRING savhelpstr[25];                           085
    % SUBSYSTEM NAME STRING %                                          086
      DECLARE EXTERNAL STRING ssysname[30];                             087
    % SUBSYSTEM STACK %                                               088
      % The subsystem stack (sbstack) contains a one word entry for
      each subsystem executed.  The information in the stack drives
      the supervisor control routine.  The execution functions for
      the GOTO, QUIT, and EXECUTE commands manipulate entries in
```

```
    this stack %                                                    089
    DECLARE EXTERNAL                                                090
       sbstkx=0,  % index to next entry in sbstack %                091
       sbstack[ sbstksize ];      % subsystem control stack %        092
  % SUBSYSTEM DISPATCH STACK %                                      093
    DECLARE EXTERNAL                                                094
       sdptx = 0,          % index to next cell in nlssubs %         095
       nlssubs[ sdptsize ], % attached subsystems dispatch stack %
                                                                    096
       sdptxa = 0,         % index to next cell in allsubs %         0222
       allsubs[ sdptsize ]; % all subsystems dispatch stack %       0223
  % RECOGNITION STATE VARIABLES %                                   097
    DECLARE EXTERNAL                                                098
       cmdmode,   % command recognition flag %                      099
       nofail=1,  % TRUE if recognizers cannot fail %               0100
       fbstr;     % ptr to current feedback string %               0101
          % this pointer is kept in a global to avoid having to
          pass it around as a actual argument among the feedback
          utility routines.  Note that the actual feedback string
          resides in the stack frame.  fbptr must be reconstructed
          when popping states so it reflects the "current"
          feedback string %                                         0102
  % REPLACE PARSE RULE SAVE AREA %                                 0103
    DECLARE EXTERNAL                                                0104
       prsavx = 0,         % index to next cell in prsavearea %     0105
       prsavearea[prsvsize];      % save area for replaced
       interpreter instructions %                                   0106
  % COMMAND COMPLETION CODE %                                       0107
    DECLARE EXTERNAL                                                0108
       compicode = 1;                                               0109
  % ACTION DATA BLOCKS %                                           0110
    DECLARE EXTERNAL                                                0111
       defaction[2],% record for processing default chars %         0112
       caaction[2],        % record for processing ca char %         0113
       optaction[2];       % record for processing opt char %        0114
  % FLAGS %                                                        0115
    DECLARE EXTERNAL                                                0116
       slink = 0,          % TRUE when doing a link/filelink
       selection %                                                  0150
       nwlnk = 0,          % TRUE when doing a new filelink
       selection %                                                  0153
       cdlnk = 0,          % TRUE to put comma after dir name  %
                                                                    0151
       clpsw = 0,          % TRUE means doing a password selection
       %                                                            0152
       auxinflag = 0,      % TRUE when reading input from aux
       source %                                                     0117
       basestateflag = 0,          % TRUE when interpreter is in
       base state %                                                 0118
       needconfirm=0, % flag for confirms after bugs %             0119
       lastsel;% previous selection opcode %                        0120
    DECLARE EXTERNAL                                                0121
       cueflg;    % TRUE when inprompts have been done %           0122
  % PATH STACK %                                                   0123
    % the path stack contains entries each of which are composed
    of two records.  The first record is the control record for
```

```
        the path stack entry and is used only by the interpreter to
        record interpreter state information for the entry.  The
        second record is the function state record and it is used by
        the processing function to record return values and save any
        local state associated with the function %                      0124
        % If a processing function requires more space for recording
        state information than is available in the function state
        record, then it should allocate the needed space and record a
        pointer to the allocated space in the function state record.
        Note that the function has the responsibility for recording
        its own state information and allocating and deallocating
        space as required.  The interpreter passes control to the
        functions, passing them a pointer to the function state record
        and a control word, and it is up the the function to manage
        the saving of state if required. %                              0125
                                                                        0126
        DECLARE EXTERNAL                                                0127
           pathx=0,   % next available cell in path stack %             0128
           pathstk[pssize],% storage for path stack entries %           0129
           pathbase = pathstk;        % ptr to base for this subsys %
                                                                        0130
      % EXECUTION POINTER STACK %                                       0131
        % the execution pointer stack (ptrstk) whose next position is
        indexed by ptrx contains pointers to "xecute" instructions in
        the grammar and marks the points where control has been
        transfered out of line.  This stack is external because it is
        also manipulated by the keyword recognition processor, which
        must chase down the alternate control paths in order to
        discover the actual path %                                      0132
        DECLARE EXTERNAL                                                0133
           ptrx = 0,  % index of next position in ptrstk %              0134
           ptrstk[ ptrssize ]; % stack for recording tree switches %
                                                                        0135
      % EVAL STACK %                                                   0136
        DECLARE EXTERNAL                                                0137
           evalx=0,   % next available cell in eval stack %             0138
           evalstk[evalsize];% storage for eval stack %                 0139
% HELP SUBSYSTEM INTERFACE POINTERS %                                   0140
   DECLARE EXTERNAL                                                     0141
      hlpcmdstk = 0,   % ptr to array with command path to be shown to
      user; 0 if array not allocated %                                  0142
      hpcmdmax = 50;   % maximum number of words in path %              0143
% jump return rings data%                                              0144
   DECLARE EXTERNAL                                                     0145
      frrcnt,  %used for jump file return%                              0146
      srrcnt; %used for jump return%                                    0147
FINISH of pdata                                                         0148
```

(MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PD
(MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PDATA (MLK)PD

```
< NLS, PDATA.NLS;13, >, 25-OCT-74 09:52 KJM ;;;;
FILE pdata  %  L10  <RFL-NLS>PDATA  %% (L10,) (rel-nls,PDATA.rel,) %
                                                                    02
% CONSTANTS %                                                      0154
   % SIZE DECLARATIONS %                                           0155
      SET EXTERNAL                                                  0156
         prsvsize = 30,          % room for 10 parse rule replacements %
                                                                    0212
         funrecsize = 10,        % size of funstaterec %           0157
         pathrecsize = 3,        % size of pathsr %                0158
         totalrecsize = 13,      % total rec size %                0159
         psdepth = 200,          % total  entries in path stack%   0160
         pssize = 2600,          % psdepth * totalrecsize %        0161
         selstatesize = 3,       % size of selstate record %       0162
         ptrssize = 20,          % size of ptrstk %                0163
            % CHANGE DECLARATIONS IN PARSER ALSO %                 0213
               % use following pattern to find spots to change    0215
                  /"ptrssize (in PDATA)"/;   %                     0216
         evalsize = 40,          % size of eval stack %            0164
         sbstksize = 20,         % size of sbstack %               0165
         sbentsize = 2,          % size of sbentry record %        0166
         framesize = 4,          % size of subsystem definition frame %
                                                                    0167
         sdptsize = 84,          % size of nissubs and allsubs %   0168
         fbstrmax = 50,          % max size of feedback string %   0169
         wasize = 15;            % work area word size (5+fbstrmax/5) %
                                                                    0170
            % CHANGE DECLARATIONS IN PARSER ALSO %                 0218
               % use following pattern to find spots to change    0219
                  /"wasize (in PDATA)"/;   %                       0220
   % VALIDATION CODE FOR SUBSYSTEM DISPATCH RECORD %               0172
      % This 18 bit validation code is output by the CML compiler and
      checked by dfnsubsys routine to make sure that the pointer being
      handed that routine actually points to a subsystem dispatch
      record %                                                      0173
      SET EXTERNAL dptvldationcode = 110473;                        0174
   % KEYWORD DELIMITER CHARS %                                     0175
      SET EXTERNAL                                                  0176
         eschar = 140B,          % ESC for recognition purposes %  0177
         leftdelim =SP;          % left delimiter for keywords %   0178
   % INTERPRETER EVALUATION MODES %                                0179
      SET EXTERNAL                                                  0180
         unknown = 0,            % dont know what type it is %     0181
         parallel = 1,           % parallel recognition %          0182
         serial = 2,             % serial recognition %            0183
         parsefunction = 3;      % interpretted by parsing function % 0184
   % INTERPRETER OPCODE DECLARATIONS %                             0185
      SET EXTERNAL                                                  0186
         % RECOGNIZERS %                                          0187
            keyop=1B,            % keyword recognition operator %  0188
            confirm=2B,          % get command confirmation %      0189
            ssel=3B,             % source selection %              0190
            dsel=4B,             % destination selection %         0191
            lsel=5B,             % literal selection %             0192
            pca=6B,              % ca recognizer %                 0193
            vwspecs=7B,          % gets viewspecs %                0194
```

```
        levadj=10B,          % get level adjust string %          0195
      % OPTIONAL ELEMENTS %                                       0196
        option=11B,          % optional parameter %               0197
        anyof=12B,           % repeated list with failure %       0198
      % CONTROL ELEMENTS %                                        0199
        pfcall=21B,          % parse function call %              0200
        xecute=22B,          % transfer to another point in tree %
                                                                  0201
        call=23B,            % subroutine call %                  0202
      % FEEDBACK ELEMENTS %                                       0203
        fbclear=31B,         % clear feedback buffer %            0204
        necho=32B,           % echo noise word string %           0205
        rechc=33B,           % replace last thing echoed %        0206
      % VALUE MANIPULATIONS %                                     0207
        store=41B,           % store value into variable %        0208
        plcad=42B,                 % load variable to ptr stack %
                                                                  0209
        enter=43B,           % enter constant into stack %        0210
        valueof=44B;         % valueof built in function %        0211
% parser flags %                                                    03
  DECLARE EXTERNAL                                                04
  littakedown = 1;     % TRUE IFF literal is to be taken down (see
  <select,getlit> %                                               05
% repeat command state info %                                     06
  DECLARE EXTERNAL STRING                                         07
    keyinpstr[50];   % actual chars typed to get recognition %    08
  DECLARE EXTERNAL STRING                                         09
    keyrptstr[50];   % recognition string at end of last command % 010
  DECLARE EXTERNAL                                                011
    keysaveflag = 1;         % TRUE if keyword inpt chars are to be
    saved %                                                       012
% parser variables %                                             013
  DECLARE EXTERNAL                                                014
    %directory and archive directory commands%                    015
      gparam,                                                     016
      gparm2,                                                     017
      gparm3,                                                     018
      gparm4,                                                     019
    aheadfilename,                                                020
    dent,                                                         021
    dest,                                                         022
    ent,                                                          023
    filtre,                                                       024
    namfil,                                                       025
    ff,                                                           026
    fromwhom,                                                     027
    level,                                                        028
    literal,                                                      029
    param,                                                        030
    param1,                                                       031
    param2,                                                       032
    param3,                                                       033
    param4,                                                      0149
    pb,                                                           034
    port,                                                         035
    retfilename,                                                  036
```

```
    s1,                                                                 037
    s2,                                                                 038
    sent,                                                               039
    sim,                                                                040
    source,                                                             041
    vs;                                                                 043
% writeable data for parser %                                          044
  % INTERPRETER CONTROL VARIABLES %                                     045
    % CURRENT MARKER %                                                 054
      DECLARE EXTERNAL TEXT POINTER                                     055
        curmkr;    % current marker %                                   056
      DECLARE EXTERNAL                                                  057
        cspupdate,          %flag to set up csp from curmkr in dnls
        -- contains da address or zero%                                058
        usesrr,    %address of statement return ring to use for
        jump file return or zero%                                       059
        cspcacode, %content analyzer code address -- see CMDFINISH%
                                                                        060
        cspusqcod, %user sequence generator program address -- see
        CMDFINISH%                                                      061
        cspvs[2];  %viewspecs to be used with curmkr if changecsp
        is set%                                                         062
    % PTR TO THE INPUT ROUTINE %                                       063
      DECLARE EXTERNAL                                                  064
        inpt;                                                           065
    % CURRENT GRAMMAR %                                                066
      DECLARE EXTERNAL                                                  067
        grammar;   % ptr to start of grammar %                         068
    % FRAME COUNTER %                                                  069
      DECLARE EXTERNAL                                                  070
        framecounter = 0;                                               071
    % CUT BACK POINTER %                                               0224
      DECLARE EXTERNAL                                                  0225
        cutstop = 0;                                                    0226
    % TERMINATION CHARACTERS FOR NLS COMMANDS %                        072
      DECLARE EXTERNAL                                                  073
        cachar=4,           % ↑D %                                      074
        optchar=25B,        % ↑U %                                      075
        rptchar=140B,       % ↑@ %                                      076
        inschar=5;                  % ↑E %                            077
  % INTERPTETER STATE DATA %                                           078
    % PROMPTS STRING %                                                 079
      DECLARE EXTERNAL STRING                                           080
        prompstr[25];                                                   081
    % COMMAND HEARALD STRING %                                         082
      DECLARE EXTERNAL STRING nrldstr[30];  % TNLS hearald string %
                                                                        083
    % STRING FOR PROMPTS FOR PFCALL FUNCTIONS %                       084
      DECLARE EXTERNAL STRING savhelpstr[25];                           085
    % SUBSYSTEM NAME STRING %                                          086
      DECLARE EXTERNAL STRING ssysname[30];                             087
    % SUBSYSTEM STACK %                                               088
      % The subsystem stack (sbstack) contains a one word entry for
      each subsystem executed.  The information in the stack drives
      the supervisor control routine.  The execution functions for
      the GOTO, QUIT, and EXECUTE commands manipulate entries in
```

```
    this stack %                                                    089
    DECLARE EXTERNAL                                                090
       sbstkx=0,  % index to next entry in sbstack %                091
       sbstack[ sbstksize ];      % subsystem control stack %        092
  % SUBSYSTEM DISPATCH STACK %                                      093
    DECLARE EXTERNAL                                                094
       sdptx = 0,          % index to next cell in nlssubs %         095
       nlssubs[ sdptsize ], % attached subsystems dispatch stack %
                                                                    096
       sdptxa = 0,         % index to next cell in allsubs %        0222
       allsubs[ sdptsize ]; % all subsystems dispatch stack %      0223
  % RECOGNITION STATE VARIABLES %                                   097
    DECLARE EXTERNAL                                                098
       cmdmode,    % command recognition flag %                    099
       nofail=1,  % TRUE if recognizers cannot fail %             0100
       fbstr;      % ptr to current feedback string %             0101
          % this pointer is kept in a global to avoid having to
          pass it around as a actual argument among the feedback
          utility routines.  Note that the actual feedback string
          resides in the stack frame.  fbptr must be reconstructed
          when popping states so it reflects the "current"
          feedback string %                                         0102
  % REPLACE PARSE RULE SAVE AREA %                                 0103
    DECLARE EXTERNAL                                                0104
       prsavx = 0,         % index to next cell in prsavearea %    0105
       prsavearea[prsvsize];       % save area for replaced
       interpreter instructions %                                   0106
  % COMMAND COMPLETION CODE %                                      0107
    DECLARE EXTERNAL                                                0108
       complcode = 1;                                               0109
  % ACTION DATA BLOCKS %                                           0110
    DECLARE EXTERNAL                                                0111
       defaction[2],% record for processing default chars %         0112
       caaction[2],        % record for processing ca char %        0113
       optaction[2];       % record for processing opt char %       0114
  % FLAGS %                                                        0115
    DECLARE EXTERNAL                                                0116
       slink = 0,          % TRUE when doing a link/filelink
       selection %                                                  0150
       nwlnk = 0,          % TRUE when doing a new filelink
       selection %                                                  0153
       cdlnk = 0,          % TRUE to put comma after dir name  %
                                                                    0151
       clpsw = 0,          % TRUE means doing a password selection
       %                                                            0152
       auxinflag = 0,      % TRUE when reading input from aux
       source %                                                     0117
       basestateflag = 0,          % TRUE when interpreter is in
       base state %                                                 0118
       needconfirm=0, % flag for confirms after bugs %             0119
       lastsel;% previous selection opcode %                        0120
    DECLARE EXTERNAL                                                0121
       cueflg;     % TRUE when inprompts have been done %          0122
  % PATH STACK %                                                   0123
    % the path stack contains entries each of which are composed
    of two records.  The first record is the control record for
```

```
        the path stack entry and is used only by the interpreter to
        record interpreter state information for the entry.  The
        second record is the function state record and it is used by
        the processing function to record return values and save any
        local state associated with the function %                      0124
        % If a processing function requires more space for recording
        state information than is available in the function state
        record, then it should allocate the needed space and record a
        pointer to the allocated space in the function state record.
        Note that the function has the responsibility for recording
        its own state information and allocating and deallocating
        space as required.  The interpreter passes control to the
        functions, passing them a pointer to the function state record
        and a control word, and it is up the the function to manage
        the saving of state if required. %                              0125
                                                                        0126
        DECLARE EXTERNAL                                                0127
           pathx=0,    % next available cell in path stack %             0128
           pathstk[pssize],% storage for path stack entries %           0129
           pathbase = pathstk;          % ptr to base for this subsys %
                                                                        0130
      % EXECUTION POINTER STACK %                                       0131
        % the execution pointer stack (ptrstk) whose next position is
        indexed by ptrx contains pointers to "xecute" instructions in
        the grammar and marks the points where control has been
        transfered out of line.  This stack is external because it is
        also manipulated by the keyword recognition processor, which
        must chase down the alternate control paths in order to
        discover the actual path %                                      0132
        DECLARE EXTERNAL                                                0133
           ptrx = 0,  % index of next position in ptrstk %              0134
           ptrstk[ ptrssize ]; % stack for recording tree switches %
                                                                        0135
      % EVAL STACK %                                                   0136
        DECLARE EXTERNAL                                                0137
           evalx=0,    % next available cell in eval stack %           0138
           evalstk[evalsize];% storage for eval stack %                 0139
% HELP SUBSYSTEM INTERFACE POINTERS %                                   0140
   DECLARE EXTERNAL                                                     0141
      hlpcmdstk = 0,    % ptr to array with command path to be shown to
      user; 0 if array not allocated %                                  0142
      hpcmdmax = 50;    % maximum number of words in path %            0143
% jump return rings data%                                              0144
   DECLARE EXTERNAL                                                     0145
      frrcnt,  %used for jump file return%                              0146
      srrcnt; %used for jump return%                                    0147
FINISH of pdata                                                         0148
```

PRMSPC

PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC
PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC PRMSPC

```
< NLS, PRMSPC.NLS;70, >, 13-NOV-74 13:36 EKM ;;;;
FILE prmspc  %  L10  <REL-NLS>PRMSPC  %% (L10,) (rel-nls,PRMSPC.rel,) %

%variable declarations%
   REF rawchr, tda;
   DECLARE
      name=1, word=2, contnt=3, sid=4, gotchr=0  getchr=1,
      frstchr=1;
   REGISTER al = 12, m = 10;

%INPUT support routines%

   (inbug) PROCEDURE(bg);
      an();
      arm();
      CASE lookc() OF
         =CD:
            BEGIN
            input();
            GOTO STATE;
            END;
         =CA, =C.:
            BEGIN
            input();
            strbug(bg);
            RETURN ;
            END;
         =BC:
            BEGIN
            input();
            BUMP [m];
            RETURN ;
            END;
         ENDCASE
            BEGIN
            BUMP [m], [m];
            RETURN;
            END;
      END.

   (instid) PROCEDURE(bg);
      LOCAL cnt;
      an();
      arm();
      CASE lookc() OF
         =CD:
            BEGIN
            input();
            GOTO STATE;
            END;
         =BUG, =C.:
            BEGIN
            input();
            strbug(bg);
            RETURN ;
            END;
```

```
      =BC:
         BEGIN
         input();
         BUMP [m];
         RETURN ;
         END;
      =SP:
         BEGIN
         input();
         *stn* ← NULL;
         cnt ← 0;
         [bg] ← origin;
         [bg].stfile ← lcfile();
         inname($stn);  %can return +1, +2, +3 past call%
         BUMP cnt, cnt;
         CASE cnt OF
            =2:
               BEGIN
               input();
               specreg($stn, name, bg);
               IF [bg] = endfil THEN
                  err($"No such statement encountered");
               RETURN;
               END;
            =1:
               BEGIN
               BUMP [m];
               RETURN;
               END;
            ENDCASE
               BEGIN
               BUMP [m],[m];
               RETURN;
               END;
         END;
      ENDCASE
         BEGIN
         BUMP [m], [m];
         RETURN;
         END;
   END.

(intext)
   %This routine reads literal input from a display terminal,
   appends it to an A-string, and displays it in the literal
   feedback area.  The argument passed this routine should be the
   address of the A-string to which the literal input is to be
   appended. (Note that this routine does not clear the A-string
   before it begins reading characters.)  TEXT = arbitrary number of
   characters up  to but not including a CA or Center dot. DOES A
   SKIP RETURN IF A BC OR BW IS INPUT AND THE STRING IS EMPTY%
   %---------------%

   PROCEDURE(astrng);
   REF astrng;
   &tda ← lda(); %for tabs%
```

```
    af();
    disarm();
    setlit();  %set up for literal collection%
    CASE rdlit(&astrng, 0) OF
       = 1: %CA or CDOT%
          BEGIN
          unput();
          RETURN;
          END;
       = 3: %BC or BW%
          BEGIN
          BUMP [m];
          RETURN;
          END;
       ENDCASE RETURN;
    END.

(innum)
    %This routine reads digits from the work station, appends
    them to an A-string, and displays them in the name register
     The argument is the address of the register
    into which the A-string is to be put. (Note that this
    routine does not clear the A-string before appending
    characters to it.).%
    %---------------%
    PROCEDURE(astrng);
    LOCAL char;
    REF astrng;
    af();
    disarm();
    LOOP
       BEGIN
       dn(&astrng);
       CASE char ← lookc() OF
          = '-:
             IF astrng.L = empty THEN
                *astrng* ← *astrng*, char
             ELSE
                BEGIN
                BUMP [m], [m];
                RETURN;
                END;
          =CA, =C.:
             BEGIN
             IF astrng.L = empty THEN
                BEGIN
                input();
                strbug($b1);
                <TXTEDT, ndr>($b1, $p1, $p2);
                *astrng* ← p1 p2;
                END;
             RETURN;
             END;
          =CD:
             BEGIN
             input();
```

```
            an();
            GOTO STATE;
            END;
         =D:
            IF astrng.L >= astrng.M THEN
               dismes(2, $"number too long -- last character(s)
               lost")
            ELSE *astrng* ← *astrng*, char;
         =BC:
            IF astrng.L = empty THEN
               BEGIN
               input();
               BUMP [m];
               RETURN;
               END
            ELSE <INPFBK, bkc>(&astrng);
         =BW:
            IF astrng.L = empty THEN
               BEGIN
               input();
               BUMP [m];
               RETURP;
               END
            ELSE <INPFBK, bkw>(&astrng);
         ENDCASE
            BEGIN
            BUMP [m], [m];
            RETURN;
            END;
      input();
      END;
   END.

(insr)
   %This routine reads characters from the work station,
   appends them to an A-string, and displays them in the name
   register.  The argument should be the address of the
   register into which the A-string is to be put. (Note that
   this routine does not clear the A-string before appending
   characters to it.)%
   %---------------%
   PROCEDURE(astrng);
   LOCAL char;
   REF astrng;
   af();
   disarm();
   LOOP
      BEGIN
      dn(&astrng);
      CASE char ← lookc() OF
         =CD:
            BEGIN
            input();
            GOTO STATE;
            END;
         =CA:
```

```
            BEGIN
            IF astrng.L = empty THEN incont(&astrng);
            RETURN;
            END;
         =C.:
            RETURN;
         =BC:
            IF astrng.L = empty THEN
               BEGIN
               input();
               BUMP [m];
               RETURN;
               END
            ELSE <INPFBK, bkc>(&astrng);
         =BW:
            IF astrng.L = empty THEN
               BEGIN
               input();
               BUMP [m];
               RETURN;
               END
            ELSE <INPFBK, bkw>(&astrng);
         ENDCASE
            IF astrng.L >= astrng.M THEN
               dismes(2, $"name too long -- last character(s) lost")
            ELSE
               *astrng* ← *astrng*, char;
      input();
      END;
   END.

(incont) PROCEDURE(string);
   %puts bugged text entity in string passed it%
   LOCAL TEXT POINTER bug1, bug2, ptr1, ptr2;
   REF string;
   INPUT BUG bug1 BUG bug2;
   <TXTEDT, tdr> ($bug1, $bug2, $ptr1, $ptr2);
   *string* ← ptr1 ptr2;
   dn(&string);
   RETURN;
   END.


(invsbl) PROCEDURE(astrng);
   CASE innmwd(astrng, FALSE, $vdr) OF
      =0:
         RETURN;
      =1:
         BEGIN
         BUMP [m];
         RETURN;
         END;
      ENDCASE
         BEGIN
         BUMP [m],[m];
         RETURN;
```

```
        END;
    END.

(inword) PROCEDURE(astrng);
    CASE innmwd(astrng, TRUE, $nmdr) OF
        =0:
            RETURN;
        =1:
            BEGIN
            BUMP [m];
            RETURN;
            END;
        ENDCASE
            BEGIN
            BUMP [m],[m];
            RETURN;
            END;
        END.

(inname) PROCEDURE(astrng);
    CASE innmwd(astrng, FALSE, $nmdr) OF
        =0:
            RETURN;
        =1:
            BEGIN
            BUMP [m];
            RETURN;
            END;
        ENDCASE
            BEGIN
            BUMP [m],[m];
            RETURN;
            END;
        END.

(innmwd)
    %This routine reads characters from the work station, appends
    them to a register, and displays them in the nameregister.
    Alphabetic characters are forced to upper case before insertion
    into the A-string if the wordflag is false.  The argument should
    be the address of the register into which the A-string is to be
    put. (Note that this routine does not clear the A-string before
    appending characters to it.).%
    %---------------%
    PROCEDURE(astrng, wordflag, delimproc);
    LOCAL char;
    REF astrng, delimproc;
    af();
    disarm();
    LOOP
        BEGIN
        dn(&astrng);
        CASE char ← lookc() OF
            =CD:
                BEGIN
                input();
```

```
            GOTO STATE;
            END;
         =CA, =C.:
            BEGIN
            IF astrng.L = empty THEN
               BEGIN
               input();
               strbug($b1);
               delimproc($b1, $p1, $p2);
               IF wordflag THEN *astrng* ← p1 p2
               ELSE *astrng* ← +p1 p2;
               dn(&astrng);
               END;
            RETURN(0);
            END;
         =BC:
            IF astrng.L = empty THEN
               BEGIN
               input();
               RETURN(1);
               END
            ELSE <INPFBK  bkc>(&astrng);
         =BW:
            IF astrng.L = empty THEN
               BEGIN
               input();
               RETURN(1);
               END
            ELSE <INPFBK, bkw>(&astrng);
         ENDCASE
            BEGIN
            IF NOT wordflag AND char IN ['a,'z] THEN
               char ← char - 40B;
            IF astrng.L >= astrng.M THEN
               dismes(2, $"name too long -- last character(s) lost")
            ELSE
               *astrng* ← *astrng*, char;
            END;
      input();
      END;
   END.

(inch) PROCEDURE(char);
   LOCAL nchar;
   af();
   disarm();
   CASE nchar ← lookc() OF
      =char:
         BEGIN
         input();
         RETURN;
         END;
      =CD:
         BEGIN
         input();
         GOTO STATE;
```

```
            END:
         =BC:
            BEGIN
            input():
            BUMP [m];
            RETURN ;
            END;
         ENDCASE
            BEGIN
            BUMP [m],[m];
            RETURN;
            END:
   END.

(oldanswer) %this procedure accepts a yes or no answer from the
keyboard, and returns with a 0 if the answer was negative, and a 1
if it was positive%
   %------------------%
   PROCEDURE:
   CASE nlmode OF
      = fulldisplay:
         CASE inpcuc() OF
            = 'Y, = CA, = SP:
               BEGIN
               DSP(?OFF Yes);
               RETURN(TRUE);
               END;
            = 'N:
               BEGIN
               DSP(?OFF No);
               RETURN(FALSE)
               END:
            = CD: GOTO STATE;
            ENDCASE
               BEGIN
               qm();
               REPEAT;
               END;
      = typewriter:
         BEGIN
         echoff();
         CASE inpcuc() OF
            = 'Y, =SP, = CA:
               BEGIN
               echo($"Yes ");
               curchr ← 'Y;
               RETURN(TRUE);
               END:
            = 'N:
               BEGIN
               echo($"No ");
               curchr ← 'N;
               RETURN(FALSE)
               END;
            = CD: GOTO STATE;
            = '?:
```

```
                BEGIN
                typeas($"
                Type CD to abort; 'y, SP, or CA for YES; 'n for NO:
                ");
                REPEAT;
                END;
             ENDCASE
                BEGIN
                typeas($" ?? ");
                REPEAT;
                END;
          END;
       ENDCASE err($"Illegal parsing mode detected in ANSWER");
    END.

(infilename) PROCEDURE (string);  %input a file name (DNLS)%
    LOCAL bugf;
    LOCAL TEXT POINTER bug, start, end;
    REF string;
    LOOP
       BEGIN
       INPUT (BUG bug bugf ← TRUE; / VISIBLE string bugf ← FALSE;);
       IF bugf THEN
          BEGIN
          flndr($bug, $start, $end);
          *string* ← start end;
          dn(&string);
          END;
       CASE input() OF
          = BC:
             BEGIN
             string.L ← MAX(string.L-1, 0);
             INPUT VISIBLE string;
             REPEAT CASE;
             END;
          = CA : RETURN;
          = CD : GOTO STATE;
          ENDCASE
             BEGIN
             IF bugf THEN delbm();
             *string* ← NULL;
             dn(&string);
             END;
       END;
    END.
(invspc) PROCEDURE (srcdpa);
    %This routine reads characters from the work station and changes
    viewspecs.  The original viewspecs are taken from the appropriate
    display area and stored in sysvspec.  The viewspecs changed are
    those in sysvspec.  That's where the new viewspecs will be on
    return..  It sets the viewspecs large upon entry, and small upon
    exit.
    A CD does a GOTO STATE.  A CA or Centerdot causes a return of the
    display area where the cursor is.  No need to call dpset.%
    %---------------%
    LOCAL da, save, sav1, sav2;
```

```
    REF srcdpa, da;
    savvspc (&srcdpa);   %move viewspecs to sysvspec%
    dspvsp (sysvspec, sysvspec[1], 3);   %display them%
    ltl ();
    af ();
    RESET savevspec;
    *vspstr* ← NULL;
    CASE input () OF
       =CD:
          BEGIN
          savvspc (&srcdpa);
          lts ();
          dn($"");
          an ();
          GOTO STATE;
          END;
       =CA, =C.:
          BEGIN
          lts ();
          an ();
          dn($"");
          RETURN (lda ());
          END;
       ='f: %recreate the display%
          BEGIN
          &da ← lda ();
          IF sysvspec.vsrlev THEN
             sysvspec ← <SEQGEN, reslev>(sysvspec, getlev(da.dacsp));
          save ← da.davspec.vsdaft := TRUE;
          sav1 ← da.davspec := sysvspec;
          sav2 ← da.davspc2 := sysvspec[1];
          dafrmt (&da, 0);
          da.davspec.vsdaft ← save;
          da.davspec ← sav1;
          da.davspc2 ← sav2;
          REPEAT CASE;
          END
       ENDCASE %includes BC%
          BEGIN
          IF curchr = BC AND  vspstr.L = empty THEN RETURN(0);
          sysvspec ←
             stklt (curchr, sysvspec, sysvspec[1] : sysvspec[1]);
          dn($vspstr);
          dspvsp (sysvspec, sysvspec[1], 3);
          REPEAT CASE;
          END;
    END.

(inlevadj) PROCEDURE(astrng);
    %this procedure gets characters for LEVADJ.  It collects
    characters ontil it encounters something other than a U or D (or
    backspace).  It puts these characters into the string passed it
    and displays them in the name register.  Note that the string
    will be empty upon RETURN if the user does no level adjust.%
    %--------------%
    LOCAL char;
```

```
    REF astrng;
    *astrng* ← NULL;
    LOOP
       BEGIN
       dn(&astrng);
       CASE (char ← lookc()) OF
          =CA, =C.:
             BEGIN
             EXIT LOOP;
             END;
          =SP:
             BEGIN
             input();
             EXIT LOOP;
             END;
          ='u, ='d:
             *astrng* ← *astrng*, char;
          =BC:
             IF astrng.L > empty THEN BUMP DOWN astrng.L
             ELSE
                BEGIN
                input();
                BUMP [m];
                EXIT LOOP;
                END;
          =BW, =$ascbst: *astrng* ← NULL;
          =CD:
             BEGIN
             input();
             GOTO STATE;
             END;
          ENDCASE BEGIN
             resetb( &astrng ); % put collected chars back into input
             buffer %
             *astrng* ← NULL;
             dn( &astrng );  % clear name buffer %
             EXIT LOOP;
             END;
       input();
       END;
    RETURN;
    END.

%input BUG's CA%

    (in1ca) PROC(bg1);
       LOCAL cdot;
       REF bg1;
       INPUT BUG bg1 (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in2ca) PROC(bg1, bg2):
       LOCAL cdot;
       REF bg1, bg2;
       INPUT BUG bg1 BUG bg2 (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.
```

```
    (in3ca) PROC(bg1, bg2, bg3);
       LOCAL cdot;
       REF bg1, bg2, bg3;
       INPUT BUG bg1 BUG bg2 BUG bg3 (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in4ca) PROC(bg1, bg2, bg3, bg4);
       LOCAL cdot;
       REF bg1, bg2, bg3, bg4;
       INPUT BUG bg1 BUG bg2 BUG bg3 BUG bg4
          (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

%input BUG's TEXT CA%

    (inotca) PROC(astrng);
       LOCAL cdot;
       REF astrng;
       INPUT TEXT astrng (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in1tca) PROC(bg1, astrng);
       LOCAL cdot;
       REF bg1, astrng;
       INPUT BUG bg1 TEXT astrng (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in2tca) PROC(bg1, bg2, astrng);
       LOCAL cdot;
       REF bg1, bg2, astrng;
       INPUT BUG bg1 BUG bg2 TEXT astrng (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in3tca) PROC(bg1, bg2, bg3, astrng);
       LOCAL cdot;
       REF bg1, bg2, bg3, astrng;
       INPUT BUG bg1 BUG bg2 BUG bg3 TEXT astrng
          (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

%input STID's CA%

    (in1sca) PROC(bg1);
       LOCAL cdot;
       REF bg1;
       INPUT STID bg1 (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in2sca) PROC(bg1, bg2);
       LOCAL cdot;
       REF bg1, bg2;
       INPUT STID bg1 STID bg2 (CA cdot ← 0; / C. cdot ← 1);
       RETURN (cdot) END.

    (in3sca) PROC(bg1, bg2, bg3);
```

```
    LOCAL cdot;
    REF bg1, bg2, bg3;
    INPUT STID bg1 STID bg2 STID bg3
       (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

(in4sca) PROC(bg1, bg2, bg3, bg4);
    LOCAL cdot;
    REF bg1, bg2, bg3, bg4;
    INPUT STID bg1 STID bg2 STID bg3 STID bg4
       (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

%input STID's TEXT CA%

(inostca) PROC(astrng);
    LOCAL cdot;
    REF astrng;
    INPUT TEXT astrng (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

(in1stca) PROC(bg1, astrng);
    LOCAL cdot;
    REF bg1, astrng;
    INPUT STID bg1 TEXT astrng (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

(in2stca) PROC(bg1, bg2, astrng<;
    LOCAL cdot;
    REF bg1, bg2, astrng;
    INPUT STID bg1 STID bg2 TEXT astrng
       (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

(in3stca) PROC(bg1, bg2, bg3, astrng);
    LOCAL cdot;
    REF bg1, bg2, bg3, astrng;
    INPUT STID bg1 STID bg2 STID bg3 TEXT astrng
       (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

%input STID's LEVADJ%

(in1sadj) PROC(bg1, astrng);
    LOCAL cdot;
    REF bg1, astrng;
    *astrng* ← NULL;
    INPUT STID bg1 LEVADJ astrng
       (CA cdot ← 0; / C. cdot ← 1);
    RETURN (cdot) END.

(in2sadj) PROC(bg1, bg2, astrng);
    LOCAL cdot;
    REF bg1, bg2, astrng;
    *astrng* ← NULL;
    INPUT STID bg1 STID bg2 LEVADJ astrng
```

```
      (CA cdot ← 0; / C. cdot ← 1);
   RETURN (cdot) END.

(in3sadj) PROC(bg1, bg2, bg3, astrng);
   LOCAL cdot;
   REF bg1, bg2, bg3, astrng;
   *astrng* ← NULL;
   INPUT STID bg1 STID bg2 STID bg3 LEVADJ astrng
      (CA cdot ← 0; / C. cdot ← 1);
   RETURN (cdot) END.

%viewspec acceptor%

(savvspc) PROCEDURE(dpa);
   %Given the address of a display area, this routine will save the
   viewspecs associated with the area in sysvspec and sysvspec[1]
   (for massaging by setlt).%
   %---------------%
   REF dpa;
   sysvspec ← dpa.davspec;
   sysvspec[1] ← dpa.davspc2;
   RETURN;
   END.

(putvspc) PROCEDURE(dpa);
   %Given the address of a display area, this routine will put the
   viewspecs saved in sysvspec and sysvspec[1] in the display area.%
   %---------------%
   REF dpa;
   (dpa.dapvs, dpa.dapvs2) ← (dpa.davspec, dpa.davspc2);%update
   previous viewspec indicators%
   dpa.davspec ← sysvspec;
   dpa.davspc2 ← sysvspec[1];
   RETURN;
   END.

(stklt) PROCEDURE(setchr, vs1, vs2);
   %Puts input viewspec characters on stack, appends them to string
   vspstr, and calls setlt to activate them.%
   %---------------%
   LOCAL vspec[2];
   vspec ← vs1;
   vspec[1] ← vs2;
   IF setchr = BC THEN
      BEGIN
      bkc($vspstr);
      POP savevspec TO vspec;
      RETURN(vspec, vspec[1]);
      END
   ELSE PUSH vspec ON savevspec;
   vspec ← setlt(setchr, vs1, vs2:vspec[1]);
   *vspstr* ← *vspstr*, setchr;
   RETURN(vspec, vspec[1]);
   END.

(feedlt)  PROCEDURE(dpa, astrng);
```

```
   %Given the address of a display area, this routine changes
   its vspecs in accord with the specificaations in the
   A-string passed it.  It passes the characters in the
   A-string to <PRMSPC, SETLT>, except for content analyzer
   patterns.%
   %------------%
   LOCAL count, length, char, vs1, vs2;
   LOCAL TEXT POINTER tp;
   LOCAL STRING castng[250];
   REF dpa, astrng;
   length ← astrng.L;
   count ← empty - 1;
   vs1 ← dpa.davspec;
   vs2 ← dpa.davspc2;
   UNTIL (count ← count+1) > length DO
      IF (char ← *astrng*[count]) = '; THEN
         BEGIN
         *castng* ← NULL;
         UNTIL (char ← *astrng*[count ← count + 1]) = '; DO
            BEGIN
            *castng* ← *castng*, char;
            IF count >= length THEN EXIT LOOP1;
            END;
         *castng* ← *castng*, " ;;";
         FIND SF(*castng*) ↑tp;
         dpa.dacacode ← cpconan ($tp, &dpa);
         END
      ELSE vs1 ← setlt(char, vs1, vs2 : vs2);
   dpa.davspec ← vs1;
   dpa.davspc2 ← vs2;
   RETURN;
   END.

(setlt)
   %This routine adjusts the viewspecs in accord with characters
   entered during view specification.  Saves the viewspec words on
   the stack savevspecfor each character input.  When a BC is input
   the stack is poped.  The strring being displayed in the name area
   is updated accordingly.%
   %---------------%
   PROCEDURE(setchr, vs1, vs2);
   LOCAL goodvs, settmp, vspec[2];
   vspec ← vs1;
   vspec[1] ← vs2;
   goodvs ← TRUE;
   CASE setchr OF
      ='a: % 1←1-1 %
         IF vspec.vsrlev THEN
            BEGIN
            IF vspec.vslev = 0 OR vspec.vslevd THEN
               BEGIN
               BUMP vspec.vslev;
               vspec.vslevd ← TRUE;
               END
            ELSE BUMP DOWN vspec.vslev;
            END
```

```
      ELSE IF vspec.vslev > 0 THEN BUMP DOWN vspec.vslev;
    ='b:  % l←l+1 %
      IF vspec.vslevd THEN
         BEGIN
         BUMP DOWN vspec.vslev;
         IF vspec.vslev = 0 THEN vspec.vslevd ← FALSE;
         END
      ELSE IF vspec.vslev < 63 THEN BUMP vspec.vslev;
    ='c:  % l←all %
      BEGIN
      vspec.vslev ← 63;
      vspec.vsrlev ← vspec.vslevd ← FALSE;
      END;
    ='d:  % l←1 %
      BEGIN
      vspec.vslev ← 1;
      vspec.vsrlev ← vspec.vslevd ← FALSE;
      END;
    ='e:  % l=rel %
      BEGIN
      IF vspec.vsrlev THEN %user is already in e state, reset
      previous e%
         BEGIN
         (rstlev):
         vspec.vslev ← vspec.vsrlev;
         vspec.vsrlev ← FALSE;
         vspec.vslevd ← FALSE;
         END;
      vspec.vsrlev ← IF vspec.vslev THEN vspec.vslev ELSE TRUE;
      vspec.vslev ← FALSE;
      vspec.vslevd ← FALSE;
      END;
    ='g:  % branch only on %
      BEGIN
      vspec.vsbrof ← TRUE;
      vspec.vsplxf ← FALSE;
      END;
    ='h:  % branch only / plex only off %
      BEGIN
      vspec.vsbrof ← FALSE;
      vspec.vsplxf ← FALSE;
      END;
    ='i:  % content analyzer success %
      BEGIN
      vspec.vscapf ← TRUE;
      vspec.vscakf ← FALSE;
      END;
    ='j:  % content analyzer off %
      BEGIN
      vspec.vscapf ← FALSE;
      vspec.vscakf ← FALSE;
      END;
    ='k:  % content analyzer k flag %
      % only use content analyzer for first statement in sequence
      %
      BEGIN
```

```
            vspec.vscakf ← TRUE;
            vspec.vscapf ← FALSE;
            END;
          ='l:  % plex only on %
            BEGIN
            vspec.vsplxf ← TRUE;
            vspec.vsbrof ← FALSE;
            END;
          ='m:  vspec.vsstnf ← TRUE;  % location numbers on %
          ='n:  vspec .vsstnf ← FALSE;  % location numbers off %
          ='o:  vspec.vsfrzf ← TRUE;  % frozen on %
          ='p:  vspec.vsfrzf ← FALSE;  % frozen off %
          ='q:  % t←t-1 %
            IF vspec.vstrnc > 0 THEN BUMP DOWN vspec.vstrnc;
          ='r:  % t←t+1 %
            IF vspec.vstrnc < 63 THEN BUMP vspec.vstrnc;
          ='s:  % t←all %
            vspec.vstrnc ← 63;
          ='t:  % t←1 %
            vspec.vstrnc ← 1;
          ='u:  % display area formatter on %
            vspec.vsdaft ← TRUE;
          ='v:  % display area formatter off %
            vspec.vsdaft ← FALSE;
          ='w:  % 1=t=all %
            BEGIN
            vspec.vstrnc ← 63;
            vspec.vslev ← 63;
            vspec.vsrlev ← vspec.vslevd ← FALSE;
            END;
          ='x:  % 1=t=1 %
            BEGIN
            vspec.vstrnc ← vspec.vslev ← 1;
            vspec.vsrlev ← vspec.vslevd ← FALSE;
            END;
          ='y:  vspec.vsblkf ← TRUE;  %  blank line on %
          ='z:  vspec.vsblkf ← FALSE;  % blank line off %
          ='A:  vspec.vsindf ← TRUE;  % indenting on %
          ='B:  vspec.vsindf ← FALSE;  % indenting off %
          ='C:  vspec.vsnamf ← TRUE;  % names on %
          ='D:  vspec.vsnamf ← FALSE;  % names off %
          ='E:  vspec.vspagf ← TRUE;  % Paging on %
          ='F:  vspec.vspagf ← FALSE;  % Paging off %
          ='G:  vspec.vsstnr ← TRUE;  % statement numbers on right %
          ='H:  vspec.vsstnr ← FALSE;  % statement numbers on left %
          ='I:  vspec.vssidf ← TRUE;  % sid flag on %
          ='J:  vspec.vssidf ← FALSE;  % sid flag off %
          ='K:  vspec.vsidtf ← TRUE;  / INITIALS, date, on %
          ='L:  vspec.vsidtf ← FALSE;  % initials, date, off %
          ='O:  vspec.vsusqf ← TRUE;  % user sequence generator on %
          ='P:  vspec.vsusqf ← FALSE;  % user sequence generator off %
        ENDCASE goodvs ← FALSE;
      RETURN(vspec, vspec[1], goodvs);
      END.

(softcd)
```

```
   %This routine is generally used by the bug accepting routines. It is
   called when they encounter a backspace character while waiting for a
   bug mark. This routine goes to state if there are no lower entries
   on the stack; otherwise, it pops the return stack, uses DELBM to
   erase the most recent bug mark, and returns.%
   %---------------%
   PROCEDURE:
   delbm();
   RETURN:
   END.

(strbug) PROCEDURE(ptr);
   %this routine will store the current bug mark in the pointer passed
   it.%
   %-------------%
   REF ptr;
   IF bugreg = endfil THEN
      ptr ← pbug(lccords() : ptr[1])
   ELSE
      BEGIN
      ptr ← bugreg;
      ptr[1] ← bugreg[1];
      END:
   RETURN:
   END.

FINISH of PRMSPC L10
```

PSEDIt

(MLK)PSEDIT (MLK)PSEDIT (MLK)PSEDIT (MLK)PSEDIT (MLK)PSEDIT (M
(MLK)PSEDIT (MLK)PSEDIT (MLK)PSEDIT (MLK)PSEDIT (MLK)PSEDIT (M

```
< NLS, PSEDIT.NLS;70, >, 29-OCT-74 12:24 EKM ;;;;
FILE psedit % L10 <rel-nls>psedit %% (110,) (rel-nls,psedit.rel,) %    02
% Declarations %                                                      03
   REGISTER r1=1, r2=2, r3=3, r4=4;                                   04
   REF msgda, rawchr, inpt, tda;                                      05
   DECLARE                                                            06
      nofile = 0, noct = 0,ctcsp = 1, ctfrz = 2,                      07
      ctcpfz = 3, ctmkr = 8, ctcmk = 9, ctcfm = 11, ctlcfm = 15;      08
                                                                      09
   DECLARE EXTERNAL                                                 03799
      copyflag = 1, moveflag = 2, trnsflag = 3, deltflag = 4;       03800
% EDITOR SUBSYSTEM %                                                 010
   %append%                                                          062
      (xappend) %Execute Append Command%                             063
         PROCEDURE                                                   064
            %FORMALS%                                                065
               (result,          %result record%                     066
               parsemode,        %parsing, backup, cleanup%          067
               sourcentity,      %source entity type%                068
               source,           %source pointer%                    069
               destentity,       %destination entity type%           070
               destination,      %destination pointer%               071
               literal);         %string to insert between destination
               and source%                                            072
               REF                                                    073
                  result, sourcentity, source, destentity, destination,
                  literal;                                            074
            LOCAL adstr[40];                                       03642
         %----------------------%                                     075
         CASE parsemode OF                                           076
            = parsing:                                               077
               BEGIN                                              03582
               CASE sourcentity OF                                03584
                  = $link:                                        03585
                     BEGIN                                        03586
                     IF source.stastr THEN                        03587
                        BEGIN                                     03588
                        IF NOT FIND                               03589
                           SF(source) $(SP/TAB) ('(//'</"--") THEN  03590
                              ST source ← '<, SF(source) SE(source);
                                                                  03591
                        IF NOT FIND                               03592
                           SE(source) $(SP/TAB) (')//'>) THEN     03593
                              ST source ← SF(source) SE(source), '>;
                                                                  03594
                        END;                                      03595
                     lnkprs( &source, $adstr);                    03596
                     source ← adstr[1s];                          03597
                        source[1] ← adstr[1s+1];                  03598
                     [&source+d2sel] ← adstr[1e];                 03599
                        [&source+d2sel+1] ← adstr[1e+1];          03600
                     END;                                         03601
                  ENDCASE;                                        03602
               CASE sourcentity OF                                  078
                  = $statement:                                    079
                     BEGIN                                         080
```

```
                clist(ctlcfm, destination.stfile, source.stfile);
                dpset(dsprfst, destination, source, endfil);
                FIND SE(destination) tcurmkr;
                IF NOT source.stastr THEN
                   cappsta(destination, source, &literal,
                   &literal+d2sel)
                ELSE
                   capptex(destination, &source, &source+d2sel,
                   &literal, &literal+d2sel);
                clupdt();
                END;
             ENDCASE err(notyet);
          END;
       ENDCASE;
    RETURN(&result);
    END.

%archive%
   (xarchive) %Execute Archive Command%
      PROCEDURE
         %FORMALS%
            (result,          %result record%
            parsemode,        %parsing, backup, cleanup%
            filename,         %name of file to be archived%
            parameters);      %do/dont delete, deferred/immediate, not
            allowed%
            REF
               result, filename, parameters;
      LOCAL rhostn, arcparms, i;
      LOCAL STRING filstr[200];
      %----------------------%
      CASE parsemode OF
         = parsing:
            BEGIN
            % parse file name %
               rhostn ← lnbfls( &filename, 0, $filstr);
            % parse the parameters %
               arcparms ← i ← FALSE;
               arcparms.flarfl.fdbarc ← TRUE;
               WHILE [parameters] := [parameters] -1 DO
                  CASE [parameters + (i←i+1)] OF
                     = $delete: arcparms.flarfl.fdbadl ← FALSE;
                     = $deferred: arcparms.flarfl.fdbarc ← TRUE;
                     %= $immediate: err(notyet);%
                     = $not:
                        BEGIN
                        arcparms.flarfl.fdbarc ← FALSE;
                        arcparms.flarfl.fdbnar ← TRUE;
                        END;
                     = $prevent: arcparms.flarfl.fdbadl ← TRUE;
                     = $reset: arcparms ← FALSE;
                     ENDCASE err(notyet);
            *lit* ← NULL;
            carcfil(rhostn, $filstr, arcparms, $lit);
```

```
            IF lit.L THEN                                                   03833
               *lit* ← "The archive status of the following files
               has been changed:", EOL, *lit*                               03834
            ELSE                                                            03835
               *lit*  ← "No files' archive status changed";                 03836
            fbctl( typecalit, $lit);                                        03837
            END;                                                             0115
         ENDCASE;                                                            0116
      RETURN(&result);                                                       0117
      END.
                                                                             0118
%break%                                                                      0119
   (xbreak) %Execute Break Command%                                          0120
      PROCEDURE                                                              0121
         %FORMALS%                                                           0122
            (result,          %result record%                               0123
            parsemode,        %parsing, backup, cleanup%                    0124
            entity,           %entity type%                                 0125
            destination,      %destination pointer%                         0126
            level);           %level adjustment string%                     0127
            REF                                                              0128
               result, entity, destination, level;                           0129
         LOCAL TEXT POINTER t1, t2;                                          0130
         LOCAL STRING locstr[5];                                             0131
      %----------------%                                                     0132
      CASE parsemode OF                                                      0133
         = parsing:                                                          0134
            CASE entity OF                                                   0135
               = $statement:                                                 0136
                  BEGIN                                                      0137
                  clist(ctcmk, destination.stfile, endfil);                  0138
                  dpset(dsprfst, destination, endfil,
                  dpstp(destination));                                       0139
                  %ignore lit for now%                                       0140
                     FIND SF(*locstr*) ↑t1 ↑t2;                              0141
                  curmkr ← cbresta(&destination, level, $t1, $t2);
                                                                             0142
                  curmkr[1] ← 1; % to start of broken stmt %                 0143
                  clupdt();                                                  0144
                  END;                                                       0145
               ENDCASE err(notyet);                                          0146
         ENDCASE;                                                            0147
      RETURN(&result);                                                       0148
      END.
                                                                             0149
%copy%                                                                       0250
   (xcopy) %Execute Copy Command%                                            0251
      PROCEDURE                                                              0252
         %FORMALS%                                                           0253
            (result,          %result record%                               0254
            parsemode,        %parsing mode (parsing, backup,
            cleanup)%                                                        0255
            sourcentity,      %source entity type%                          0256
            source,           %source pointer%                              0257
            destentity,       %destination entity type%                     0258
            destination,      %destination pointer%                         0259
```

```
        level,  %level adjustment string%                          0260
        filterflag,     %if TRUE, filtered with viewspecs in vs%
                                                                   0261
        vs);            %viewspec string%                          0262
    LOCAL                                                          0263
        tlngth,                                                   03718
        rhostn, rhost2, adstr[40],                                03550
        % stuff for copy directory %                               0264
          info,         % record saying what was requested %      0265
          gropk,        % record saying how to group things %
                                                                   0266
          sortk         % record saying how to wort things %      0267
        ;                                                          0268
    LOCAL TEXT POINTER f1, f2;                                    03644
    LOCAL STRING                                                   0285
        filstr[200], filst2[200];                                  0286
    REF                                                            0287
        result, sourcentity, source, destentity, destination,
        level, filterflag, vs;                                     0288
%----------------------%                                          0289
CASE parsemode OF                                                  0290
    = parsing:                                                     0291
        BEGIN                                                     03551
        result ← 0;                                               03721
        CASE sourcentity OF                                       03553
            = $link:                                              03554
                BEGIN                                             03555
                IF source.stastr THEN                             03556
                    BEGIN                                         03557
                    IF NOT FIND                                   03558
                        SF(source) $(SP/TAB) ('(/'</"--") THEN    03559
                            ST source ← '<, SF(source) SE(source);
                                                                  03560
                    IF NOT FIND                                   03561
                        SE(source) $(SP/TAB) (')/'>) THEN         03562
                            ST source ← SF(source) SE(source), '>;
                                                                  03563
                    END;                                          03564
                lnkprs( &source, $adstr);                         03565
                source ← adstr[1s];                               03566
                    source[1] ← adstr[1s+1];                      03567
                [&source+d2sel] ← adstr[1e];                      03568
                    [&source+d2sel+1] ← adstr[1e+1];              03569
                END;                                              03570
            ENDCASE;                                              03571
        CASE sourcentity OF                                        0292
            %text/structure entities%                              0293
                = $character, = $invisible, = $text:               0294
                    BEGIN                                          0295
                    clist (ctcmk, destination.stfile,
                    source.stfile);                                0296
                    dpset(dsprfmt, destination, endfil,
                    destination);                                  0297
                    curmkr ← destination;                          0298
                    curmkr[1] ← destination[d2sel+1] +
                    source[d2sel+1] - source[1] - 1;               0299
```

```
                ccoptex(&destination+d2sel, &source,
                &source+d2sel, FALSE);                                  0300
                clupdt ();                                              0301
                END;                                                    0302
             = $word, = $visible, = $number, = $link:                   0303
                BEGIN                                                   0304
                clist (ctemk, destination.stfile,
                source.stfile);                                         0305
                dpset(dsprfmt, destination, endfil,
                destination);                                           0306
                curmkr ← destination;                                   0307
                curmkr[1] ← destination[d2sel+1] +
                source[d2sel+1] - source[1];                            0308
                ccoptex(&destination+d2sel, &source,
                &source+d2sel, TRUE);                                   0309
                clupdt ();                                              0310
                END;                                                    0311
             = $statement:                                              0312
                BEGIN                                                   0313
                curmkr ← xcmst(&destination, level, &source,
                copyflag, filterflag, &vs);                             0314
                curmkr[1] ← 1;                                          0315
                END;                                                    0316
             = $group, = $plex, = $branch:                              0317
                BEGIN                                                   0318
                curmkr ← xcmgrp(&destination, level, &source,
                copyflag, filterflag, &vs);                             0319
                curmkr[1] ← 1;                                          0320
                END;                                                    0321
          = $file:                                                      0322
             BEGIN                                                      0323
             % get and initialize message string %                      03732
                result ← getstring( 3000, $dspblk);                     03723
                *[result]* ← "Copied Files Are:
                ";                                                      03724
                tlngth ← [result].L;                                    03725
             % parse source file name %                                 0324
                rhostn ← lnbfls( &source, 0, $filstr);                  03572
             % parse destination file name %                            0326
                rhost2 ← lnbfls( &destination, 0, $filst2);
                                                                        03573
             ccopfil(rhostn, $filstr, rhost2, $filst2, result);
                                                                        0332
             % tell the user what we did %                             03733
                IF ( [result].L > tlngth ) THEN                         03728
                   fbctl( typecalit, result)                            07729
                ELSE fbctl( typecalit, $"No Files Copied");
                                                                        03730
             END;                                                       0333
          = $directory:                                                 0334
             BEGIN                                                      0335
             dpset(dspstrc,destination,endfil,endfil);                  0336
             info ← gropk ← sortk ← 0;                                  0337
             *filstr* ← "*.*;*";                                        03604
             xdiropt( &source, destentity, &filterflag, $info,
             $gropk, $sortk, $rhostn, $filstr);                         0338
```

```
                curmkr ← ccopdir(&destination, level, info,
                   gropk, sortk, rhostn, $filstr);
                curmkr[1] ← 1;
                END;
             = $archive:
                BEGIN
                curmkr ← ccoparcdir(&destination, level, &source,
                &source+d2sel, destentity);
                curmkr[1] ← 1;
                END;
             = $sequential:
                BEGIN
                dpset(dsprfst, destination, endfil,
                dpstp(destination));
                % move file name to local string %
                   CASE lnbfls( &source, 0, $filstr) OF
                      = lhostn: NULL;
                      ENDCASE
                         err($"Remote File Manipulations Not
                         Implemented Yet");
                % setup text pointers to start and end of string %

                   FIND SF(*filstr*) ↑f1 SE(*filstr*) ↑f2;
                CASE destentity OF
                   = $two: destentity ← neurfil;
                   = $justified: destentity ← justfil;
                   = $assembler: destentity ← assfil;
                   = 0: %normal% destentity ← tenfil;
                   ENDCASE err(notyet);
                curmkr ← ccopseqfil (destination, level, $f1, $f2,
                destentity);
                curmkr[1] ← 1;
                END;
             ENDCASE err(notyet);
          END;
       = backup, = cleanup:
          IF result THEN freestring(result, $dspblk);
       ENDCASE;
    RETURN(&result);
    END.

%copy/move support routines%
    (xdircpt)     % parse input for directory commands %
       PROCEDURE
          (source,        % record ptr to text ptrs for dir. name%

          dent,    % record ptr to parameter list %
          frfile,          % record ptr to filelink %
          info,    % adr: record to get request info %
          gropk,   % adr: record to get request info %
          sortk,   % adr: record to get request info %
          rhostn,          % adr: cell to get host number %
          deffil   % adr: default file string %
          );
       LOCAL
          pptr,    % temp pointer %
```

```
    count   % temp counter %                                          0388
    ;                                                                 0389
LOCAL TEXT POINTER tptr;                                             03605
LOCAL STRING tstrng[40];                                              0391
REF source, pptr, info, gropk, sortk, frfile, dent, rhostn,
deffil;                                                               0392
                                                                      0393
% set up file group string %                                         0394
   rhostn ← lhostn;                                                  03615
   IF source THEN                                                     0396
      BEGIN                                                           0397
      tptr ← [&source+d2sel];                                         0400
      tptr[1] ← [&source+d2sel+1];                                    0401
      IF FIND tptr < '<ESC> ↑tptr THEN                               03606
         *deffil* ← '<, source tptr, '<↑F>, *deffil*                 03608
      ELSE                                                           03610
         IF FIND tptr < '<↑F> THEN                                   03612
            *deffil* ← '<, source tptr, *deffil*                     03611
         ELSE                                                        03613
            *deffil* ← '<, source tptr, '>, *deffil*;                03614
      END;                                                            0402
% parse the parameter input %                                         0410
   count ← dent/4;          % each item is 4 elements long %
                                                                      0411
   &pptr ← &dent + 1;                                                 0412
   WHILE count DO                                                     0413
      BEGIN                                                           0414
      CASE pptr OF                                                    0415
         = $both: info.dlidlt ← 2;                                    0416
         = $delete: info.dlidlt ← 1;                                  0417
         = $undelete: info.dlidlt ← 0;                                0418
         = $for:                                                      0419
            BEGIN                                                     0420
            rhostn ← lnbfls( &frfile, 0, &deffil);                   03616
            IF NOT FIND SF(*deffil*) '< THEN                         03617
               IF source THEN                                        03618
                  BEGIN                                              03619
                  tptr ← [&source+d2sel];                            03620
                  tptr[1] ← [&source+d2sel+1];                       03621
                  IF FIND tptr < '<ESC> ↑tptr THEN                   03622
                     *deffil* ←                                      03623
                        '<, source tptr, '<↑F>, *deffil*
                                                                     03630
                  ELSE                                               03624
                     IF FIND tptr < '<↑F> THEN                       03625
                        *deffil* ←                                   03626
                           '<, source tptr, *deffil*                 03631
                     ELSE                                            03627
                        *deffil* ←                                   03628
                           '<, source tptr, '>, *deffil*;
                                                                     03632
                  END;                                               03629
            END;                                                      0431
         = $archive:                                                  0432
            CASE [&pptr + 1] OF                                       0433
               = $status: info.dliars ← TRUE;                         0434
```

```
        = $tape: info.dliart ← TRUE;                                    0435
          ENDCASE;                                                      0436
    = $account: info.dliacc ← TRUE;                                     0437
    = $no:                                                              03161
      CASE [&pptr + 1] OF                                               03162
        = $versions: info.dlinvr ← 1;                                   03163
        = $extension: info.dlinex ← 1;                                  03164
          ENDCASE;                                                      03166
    = $date:                                                            0438
      CASE [&pptr + 1] OF                                               0439
        = $archive: info.dlitar ← 1;                                    0440
        = $creation: info.dlitcr ← 1;                                   0441
        = $last: info.dlitdm ← 1;                                       0442
        = $first: info.dlitov ← 1;                                      0443
        = $read: info.dlitrd ← 1;                                       0444
        = $write: info.dlitwr ← 1;                                      0445
          ENDCASE;                                                      0446
    = $dump: info.dlidmt ← TRUE;                                        0447
    = $everything:                                                      0448
      BEGIN                                                             0449
      info.dliacc ← TRUE;                                               0450
      info.dliars ← TRUE;                                               0451
      info.dliart ← TRUE;                                               0452
      info.dlidmt ← TRUE;                                               0453
      info.dlidfr ← TRUE;                                               0454
      info.dlilwr ← TRUE;                                               0455
      info.dlibyt ← TRUE;                                               0456
      info.dlimis ← TRUE;                                               0457
      info.dlinrw ← TRUE;                                               0458
      info.dliprt ← TRUE;                                               0459
      info.dlisiz ← TRUE;                                               0460
      info.dlitar ← 2;                                                  0461
      info.dlitcr ← 2;                                                  0462
      info.dlitdm ← 2;                                                  0463
      info.dlitov ← 2;                                                  0464
      info.dlitrd ← 2;                                                  0465
      info.dlitwr ← 2;                                                  0466
      END;                                                              0467
    = $last: info.dlilwr ← TRUE;                                        0468
    = $length: info.dlibyt ← TRUE;                                      0469
    = $miscellaneous: info.dlimis ← TRUE;                               0470
    = $number:                                                          0471
      CASE [&pptr + 1] OF                                               0472
        = $versions: info.dlidfr ← TRUE;                                0473
        = $accesses: info.dlinrw ← TRUE;                                0474
          ENDCASE;                                                      0475
    = $protect: info.dliprt ← TRUE;                                     0476
    = $size: info.dlisiz ← TRUE;                                        0477
    = $time:                                                            0478
      CASE [&pptr + 1] OF                                               0479
        = $archive: info.dlitar ← 2;                                    0480
        = $creation: info.dlitcr ← 2;                                   0481
        = $last: info.dlitdm ← 2;                                       0482
        = $first: info.dlitov ← 2;                                      0483
        = $read: info.dlitrd ← 2;                                       0484
        = $write: info.dlitwr ← 2;                                      0485
```

```
            ENDCASE;                                                    0486
        = $verbose:                                                     0487
            BEGIN                                                       0488
            info.dlisiz ← TRUE;                                         0489
            info.dlilwr ← TRUE;                                         0490
            IF NOT info.dlitwr THEN info.dlitwr ← 1;                    0491
            IF NOT info.dlitrd THEN info.dlitrd ← 1;                    0492
            END;                                                        0493
        = $group:                                                       0494
            BEGIN                                                       0495
            gropk ← 0;                                                  0496
            IF [&pptr + 1] = $reverse THEN gropk.dlgrvr ←
            TRUE;                                                       0497
            CASE [&pptr + 2] OF                                         0498
               = $account: gropk.dlgacc ← TRUE;                         0499
               = $archive:                                              0500
                  CASE [&pptr + 3] OF                                   0501
                     = $date: gropk.dlgdar ← TRUE;                      0502
                     = $status: gropk.dlgars ← TRUE;                    0503
                     = $tape: gropk.dlgart ← TRUE;                      0504
                     ENDCASE;                                           0505
               = $creation: gropk.dlgdcr ← TRUE;                        0506
               = $delete: gropk.dlgdlt ← TRUE;                          0507
               = $dump:                                                 0508
                  CASE [&pptr + 3] OF                                   0509
                     = $date: gropk.dlgdm ← TRUE;                       0510
                     = $tape: gropk.dlgdmt ← TRUE;                      0511
                     ENDCASE;                                           0512
               = $first: gropk.dlgdcv ← TRUE;                           0513
               = $last: gropk.dlglwr ← TRUE;                            0514
               = $number: gropk.dlgdfr ← TRUE;                          0515
               = $protect: gropk.dlgprt ← TRUE;                         0516
               = $read: gropk.dlgdrd ← TRUE;                            0517
               = $write: gropk.dlgdwr ← TRUE;                           0518
               ENDCASE;                                                 0519
            END;                                                        0520
        = $sort:                                                        0521
            BEGIN                                                       0522
            sortk ← 0;                                                  0523
            IF [&pptr + 1] = $reverse THEN sortk.dlsrvr ←
            TRUE;                                                       0524
            CASE [&pptr + 2] OF                                         0525
               = $account: sortk.dlsacc ← TRUE;                         0526
               = $archive:                                              0527
                  CASE [&pptr + 3] OF                                   0528
                     = $time: sortk.dlstar ← TRUE;                      0529
                     = $tape: sortk.dlsart ← TRUE;                      0530
                     ENDCASE;                                           0531
               = $bytesize: sortk.dlsbyt ← TRUE;                        0532
               = $creation: sortk.dlstcr ← TRUE;                        0533
               = $delete: sortk.dlsdlt ← TRUE;                          0534
               = $dump:                                                 0535
                  CASE [&pptr + 3] OF                                   0536
                     = $time: sortk.dlstdm ← TRUE;                      0537
                     = $tape: sortk.dlsdmt ← TRUE;                      0538
                     ENDCASE;                                           0539
```

```
                        = $last: sortk.dlslwr ← TRUE;                    0540
                        = $length: sortk.dlslen ← TRUE;                  0541
                        = $number:                                       0542
                           CASE [&pptr + 3] OF                           0543
                              = $accesses: sortk.dlsnac ← TRUE;          0544
                              = $read: sortk.dlsnrd ← TRUE;              0545
                              = $write: sortk.dlsnwr ← TRUE;             0546
                              = $versions: sortk.dlsdfr ← TRUE;          0547
                              ENDCASE;                                   0548
                        = $first: sortk.dlstov ← TRUE;                   0549
                        = $read: sortk.dlstrd ← TRUE;                    0550
                        = $size: sortk.dlssiz ← TRUE;                    0551
                        = $write: sortk?dlstwr ← TRUE;                   0552
                        ENDCASE;                                         0553
                     END;                                                0554
                  ENDCASE;                                               0555
               BUMP DOWN count;                                          0556
               &pptr ← &pptr + 4;                                        0557
               END;                                                      0558
      % done so return %                                                 0559
         RETURN;                                                         0560
                                                                         0561
      END.                                                               0562

   (xcmst) PROCEDURE(destination, level, source, type,
   filterflag, vs);                                                      0563
      %copy or move statement%                                           0564
      LOCAL clistcalled, newsid, proc;                                   0565
      LOCAL STRING vsstr[50], levstr[50];                                0566
      REF proc, destination, source, vs;                                 0567
      &proc ← (IF type = copyflag THEN $ccopsta ELSE $cmovsta);
                                                                         0568
      IF clistcalled ← ( type NOT= copyflag AND NOT source.stastr
      AND source.stfile NOT= destination.stfile ) THEN                   0569
         clist (15, source.stfile, nofile);                             03767
      IF NOT source.stastr THEN                                          0573
         BEGIN                                                          03735
         newsid ← proc(destination, level, source, filterflag,
         &vs);                                                          0574
         IF type = copyflag THEN                                        03738
            dpset(dspstrc, newsid, endfil, getnxt(newsid))              03740
         ELSE                                                           03741
            dpset(dspstrc, source, destination, endfil);                03742
         END                                                            03736
      ELSE %do an insert%                                               0575
         BEGIN                                                          03683
         newsid ← cinssta(destination, level, &source,
         &source+d2sel);                                                0576
         dpset(dspstrc, newsid, endfil, getnxt(newsid));                03681
         END;                                                           03682
      IF clistcalled THEN clupdt ();                                    0577
      RETURN(newsid);                                                   0578
      END.                                                              0579

   (xcmgrp) PROCEDURE(destination, level, source, type,
   filterflag, vs);                                                      0580
```

```
        %copy or move group%                                                0581
        LOCAL clistcalled, proc, newsid, stid;                              0582
        LOCAL STRING vsstr[50], levstr[50];                                 0583
        REF proc, destination, source, vs;                                  0584
        &proc ← (IF type = copyflag THEN $ccopgro ELSE $cmovgro);
                                                                            0585
        IF clistcalled ← ( type NOT= copyflag AND NOT source.stastr
        AND source.stfile NOT= destination.stfile) THEN                     0586
           clist (15, source.stfile, nofile);                              03768
        IF NOT source.stastr THEN                                           0589
           BEGIN                                                           03744
           stid ← (IF type = copyflag THEN dpstp(destination) ELSE
           endfil);                                                         0587
           newsid ← proc(destination, level, source,
           [&source+d2sel], filterflag, &vs);                               0590
           dpset(dspstrc, newsid, IF type = copyflag THEN endfil
           ELSE source, stid);                                             03747
           END                                                             03746
        ELSE %do an insert%                                                 0591
           BE?IN                                                           03686
           newsid ← cinssta(destination, level, &source,
           &source+d2sel);                                                  0592
           dpset(dspstrc, newsid, endfil, getnxt(newsid));                 03684
           END;                                                            03685
        IF clistcalled THEN clupdt ();                                      0593
        RETURN(newsid);                                                     0594
        END.
                                                                            0595
%create%                                                                    0596
   (xcreate) %Execute Create Command%                                       0597
      PROCEDURE                                                             0598
         %FORMALS%                                                          0599
            (result,          %result record%                               0600
            parsemode,        %parsing, backup, cleanup%                    0601
            filename);        %name of file to create%                      0602
            REF                                                             0603
               result, filename;                                            0604
         LOCAL da, rhostn;  REF da;                                         0605
         LOCAL STRING filstr[200];                                          0606
      %----------------------%                                             0609
      CASE parsemode OF                                                     0610
         = parsing:                                                         0611
            BEGIN                                                           0612
            cspupdate ← &da ← lda();                                        0613
            % parse input file link %                                       0614
               rhostn ← lnbfls( &filename, 0, $filstr);                     0615
            curmkr ← ccrefil(rhostn, $filstr);                              0616
               %returns stid to origin%                                     0619
            curmkr[1] ← 1;                                                  0620
            dpset( dspyes, curmkr, endfil, endfil );                        0621
            cspvs ← da.davspec;                                             0622
            cspvs[1] ← da.davspc2;                                          0623
            END;                                                            0624
         ENDCASE;                                                           0625
      RETURN(&result);                                                      0626
      END.
```

```
                                                                    0627
%delete%                                                            0628
   (xdelete) %Execute Delete Command%                               0629
      PROCEDURE                                                     0630
         %FORMALS%                                                  0631
            (result,          %result record%                       0632
            parsemode,        %parsing, backup, cleanup%            0633
            entity,           %entity type%                         0634
            destination,      %destination pointer%                 0635
            filterflag,       %if TRUE, filtered with viewspecs in vs%
                                                                    0636
            vs);              %viewspec string%                     0637
            REF                                                     0638
               result, entity, destination, filterflag, vs;         0639
         LOCAL stid, type, da, deleteda, cords, tlngth, rhostn,
         adstr[40];                                                 0640
         REF da, deleteda;                                         04216
         LOCAL TEXT POINTER z1;                                    03643
         LOCAL STRING filstr[200];                                 03529
         %----------------------%                                   0643
         CASE parsemode OF                                          0644
            = parsing:                                              0645
               BEGIN                                                0646
               result ← 0;                                          0647
               CASE entity OF                                    03530
                  = $link:                                       03531
                     BEGIN                                       03532
                     lnkprs( &destination, $adstr);              03542
                     destination ← adstr[ls];                    03543
                        destination[1] ← adstr[ls+1];            03544
                     [&destination+d2sel] ← adstr[le];           03545
                        [&destination+d2sel+1] ← adstr[le+1];    03546
                     END;                                        03547
                  ENDCASE;                                       03548
               CASE entity OF                                       0648
                  %text and structure entities%                     0649
                     = $character, = $text, = $invisible:           0650
                        BEGIN                                       0651
                        clist (ctcmk, destination.stfile, nofile);  0652
                        dpset(dsprfmt, destination, endfil,
                        destination);                               0653
                        FIND destination ↑curmkr;                   0656
                        cdeltex(&destination, &destination+d2sel,
                        FALSE);                                     0657
                        IF FIND curmkr > ENDCHR THEN FIND curmkr
                        ←curmkr;                                 03805
                        clupdt ();                                  0658
                        END;                                        0659
                     = $word, = $visible, = $number, = $link:       0660
                        BEGIN                                       0661
                        clist (ctcmk, destination.stfile, nofile);  0662
                        dpset(dsprfmt, destination, endfil,
                        destination);                               0663
                        z1 ← destination[d2sel];                 03638
                           z1[1] ← destination[d2sel+1];         03639
                        FIND destination ↑curmkr;                   0666
```

```
            IF NOT FIND z1 > SP THEN                                03610
               IF FIND destination < SP ↑curmkr THEN NULL;
                                                                    03611
            cdeltex(&destination, &destination+d2sel,
            TRUE);                                                  0667
            IF FIND curmkr > ENDCHR THEN FIND curmkr
            ←curmkr;                                                03806
            clupdt ();                                              0668
            END;                                                    0669
         = $statement:                                              0670
            BEGIN                                                   0671
            clist (ctlcfm, destination.stfile, nofile);             0672
            dpset(dspstrc, destination, endfil,
            dpstp(destination));                                    0673
            CASE curmkr ← getnxt(destination) OF                    0674
               = destination,                                       0675
               = endfil:                                            0676
                  curmkr ← getbck(destination);                     0677
               ENDCASE;                                             0678
            curmkr[1] ← 1;                                          0679
            cdelsta(destination, filterflag, &vs);                  0680
            clupdt ();                                              0681
            END;                                                    0682
         = $group, = $plex, = $branch:                              0683
            BEGIN                                                   0684
            clist (ctlcfm, destination.stfile, nofile);             0685
            dpset(dspstrc, destination, endfil,
            dpstp(destination));                                    0686
            CASE curmkr ←
            getnxt(getend(destination[d2sel])) OF                   0687
               = endfil:                                            0688
                  curmkr ← getbck(destination);                     0689
               ENDCASE;                                             0690
            curmkr[1] ← 1;                                          0691
            cdelgro(destination, destination[d2sel],
            filterflag, &vs);                                       0692
            clupdt ();                                              0693
            END;                                                    0694
      = $file:                                                      0695
         BEGIN                                                      0696
         result ← getstring( 3000, $dspblk);                        0697
         *[result]* ← "Deleted Files Are:
         ";                                                         0698
         tlngth ← [result].L;                                       0699
         rhostn ← lnbfls( &destination, 0, $filstr);                03549
         cdelfil( rhostn, $filstr, result);                         0702
         IF ( [result].L > tlngth ) THEN                            0703
            fbctl( typecalit, result)                               0704
         ELSE fbctl( typecalit, $"No Files Deleted");               0705
         END;                                                       0706
      = $archived: %file%                                           0707
         cdelarcfil(&destination, &destination+d2sel);              0708
      = $marker:                                                    0709
         cdelmar(&destination, &destination+d2sel,
         lcfile());                                                 0710
      = $all: %markers%                                             0711
```

```
    cdelallmar(lcfile()); %must know file%                    0712
= $modifications: %to file%                                   0713
    BEGIN                                                     0714
    stid ← orgstid;                                           0715
    stid.stfile ← lcfile();                                   0716
    clist (ctlcfm, stid.stfile, nofile);                      0717
    dpset(dspallf, stid, endfil, endfil);                     0718
    cdelmodfil(stid.stfile);                                  0719
    <IOEXEC, unlkclist> (); %check clist items%               0720
    clupdt ();                                                0721
    END;                                                      0722
= $edge: %of window%                                          04217
    BEGIN                                                     04218
    &da ← dsparea(boundry(destination[1] :
    destination[1], type));                                   04220
    cords ← lccords();                                        04221
    dpset(dspallf, endfil, endfil, endfil);                   04290
    clearda(0);                                               04291
    clrall(0, TRUE);                                          04292
    ON SIGNAL ELSE alldsp();                                  04474
    CASE type OF                                              04222
       = lbound: %left edge of window DA to be
       deleted%                                               04252
          BEGIN                                               04253
          %da points to right window%                         04266
          IF da.daleft = taleft THEN                          04286
             err($"cannot delete margin edge");               04463
          IF NOT da.dalneighbor THEN                          04464
             err($"This window does not have a left
             neighbor");                                      04465
          &deleteda ← dsparea(da.dalneighbor); %left
          window%                                             04254
          IF cords.xcord > da.daleft THEN %keep
          windows to right of boundary%                       04255
             BEGIN                                            04256
             fixbnd(TRUE, da.daleft, deleteda.daleft,
             TRUE);                                           04455
             END                                              04258
          ELSE %keep windows to left of boundary%             04259
             BEGIN                                            04260
             &deleteda ← &da := &deleteda;                    04261
             %now da is left and deleteda is right
             window%                                          04267
             fixbnd(TRUE, da.daright,
             deleteda.daright, TRUE);                         04456
             END;                                             04263
          END;                                                04265
       = rbound: %right edge of window to be deleted%
                                                              04223
          BEGIN                                               04235
          IF da.daright = taright THEN                        04287
             err($"cannot delete margin edge");               04472
          IF NOT da.darneighbor THEN                          04466
             err($"This window does not have a right
             neighbor");                                      04467
          &deleteda ← dsparea(da.darneighbor); %right
```

```
    window%                                                    04226
    IF cords.xcord <= da.daright THEN %keep one
    to left of boundary%                                       04224
       BEGIN                                                   04225
       fixbnd(TRUE, da.daright,
       deleteda.daright, TRUE);                                04457
       END                                                     04229
    ELSE %keep one to right of boundary%                       04230
       BEGIN                                                   04231
       &deleteda ← &da := &deleteda;                           04237
       fixbnd(TRUE, da.daleft, deleteda.daleft,
       TRUE);                                                  04458
       END;                                                    04234
    END;                                                       04236
= tbound: %top edge of window to be deleted%
                                                               04238
    BEGIN                                                      04239
    IF da.datop = tatop THEN                                   04288
       err($"cannot delete margin edge");                      04475
    IF NOT da.datneighbor THEN                                 04468
       err($"This window does not have a top
       neighbor");                                             04469
    &deleteda ← dsparea(da.datneighbor); %top
    window%                                                    04240
    IF cords.ycord > da.datop THEN %keep window
    on bottom of edge%                                         04241
       BEGIN                                                   04242
       fixbnd(FALSE  da.datop, deleteda.datop,
       TRUE);                                                  04459
       END                                                     04244
    ELSE %keep window on top of edge%                          04245
       BEGIN                                                   04246
       &deleteda ← &da := &deleteda;                           04247
       fixbnd(FALSE, da.dabottom,
       deleteda.dabottom, TRUE);                               04460
       END;                                                    04249
    END;                                                       04251
= bbound: %bottom edge of window to be deleted%
                                                               04269
    BEGIN                                                      04270
    IF da.dabottom = tabottom THEN                             04289
       err($"cannot delete margin edge");                      04473
    IF NOT da.dabneighbor THEN                                 04470
       err($"This window does not have a bottom
       neighbor");                                             04471
    &deleteda ← dsparea(da.dabneighbor); %bottom
    window%                                                    04271
    IF cords.ycord <= da.dabottom THEN %keep
    window on top of edge%                                     04272
       BEGIN                                                   04273
       fixbnd(FALSE, da.dabottom,
       deleteda.dabottom, TRUE);                               04461
       END                                                     04275
    ELSE %keep window on bottom of edge%                       04276
       BEGIN                                                   04277
       &deleteda ← &da := &deleteda;                           04278
```

```
                        fixbnd(FALSE, da.datop, deleteda.datop,
                        TRUE);                                          04462
                        END;                                            04280
                     END;                                               04282
                  ENDCASE;                                              04284
               fixbuf();                                                04283
               END;                                                     04219
            ENDCASE err(notyet);                                         0723
         END;                                                            0724
      = backup, = cleanup:                                               0725
         IF result THEN freestring(result, $dspblk);                     0726
      ENDCASE;                                                           0727
   RETURN(&result);                                                      0728
   END.
                                                                         0729
%edit%                                                                   0745
   (xedit) %Execute Edit Command%                                        0746
      PROCEDURE                                                          0747
         %FORMALS%                                                       0748
            (result,                    %result record%                  0749
            parsemode,                  %parsing, backup, cleanup%       0750
            destination);   %destination pointer%                        0751
            REF                                                          0752
               result, destination;                                      0753
         LOCAL TEXT POINTER t1, t2;                                      0754
      %----------------------%                                           0755
      CASE parsemode OF                                                  0756
         = parsing:                                                      0757
            BEGIN                                                        0758
            dpset(dsprfmt, destination, endfil, endfil);                 0764
            t1 ← destination;  t1[1] ← destination[1];                  03782
            editx(@t1); %fill the global string LIT with the new
            version%                                                     0759
            FIND SF(*lit*) ↑t1 SE(*lit*) ↑t2;                            0760
            creptex(&destination, &destination+d2sel, $t1, $t2);         0761
            curmkr ← destination;                                        0762
            curmkr[1] ← 1;                                               0763
            END;                                                         0765
         ENDCASE;                                                        0766
      RETURN(&result);                                                   0767
      END.
                                                                         0768
%expunge%                                                                0769
   (xexpunge) %Execute Expunge Command%                                  0770
      PROCEDURE                                                          0771
         %FORMALS%                                                       0772
            (result,          %result record%                            0773
            parsemode,        %parsing, backup, cleanup%                 0774
            entity);          %type of directory%                        0775
            REF                                                          0776
               result, entity;                                           0777
      %----------------------%                                           0778
      CASE parsemode OF                                                  0779
         = parsing:                                                      0780
            CASE entity OF                                               0781
               = $directory:                                             0782
```

```
                % no parameters for the time being %                    0783
                cexpdir(0);  %expunge connected directory%              0784
             = $archive:                                                0785
                cexparcdir();  %expunge archive directory%              0786
             ENDCASE err(notyet);                                       0787
          ENDCASE;                                                      0788
       RETURN(&result);                                                 0789
       END.
                                                                        0790
%force%                                                                 03887
   (xforce) %Force case Set Command%                                    03888
      PROCEDURE                                                         03889
         %FORMALS%                                                      03890
            (result,          %result record%                           03891
            parsemode,        %parsing, backup, cleanup%                03892
            param1,           %parameter one%                           03894
            param2,           %parameter two%                           03895
            destination); %destination pointer%                         03897
            REF                                                         03898
               result, param1, param2, destination;                     03899
         LOCAL csize, hinc, vinc, da, end1, save, tp2, stid,
         adstr[10];                                                     03900
         LOCAL STRING sizestring[10];                                   03901
         REF da, tp2;                                                   03902
      %----------------------%                                          03903
      CASE parsemode OF                                                 03904
         = parsing:                                                     03905
            BEGIN                                                       03906
            dpset(dspno, endfil, endfil, endfil);                       03907
            param2 ← CASE param2 OF                                     03911
               =0: xsmode; % none specified %                           03912
               =$first: iupcase;                                        03913
               =$upper: upcase;                                         03914
               =$lower: lowcase;                                        03915
               ENDCASE err( notyet );                                   03916
            CASE param1 OF                                              03917
               = $link:                                                 03918
                  BEGIN                                                 03919
                  lnkprs( &destination, $adstr);                        03920
                  destination ← adstr[ls];                              03921
                     destination[1] ← adstr[ls+1];                      03922
                  [&destination+d2sel] ← adstr[le];                     03923
                     [&destination+d2sel+1] ← adstr[le+1];              03924
                  END;                                                  03925
               ENDCASE;                                                 03926
            CASE param1 OF                                              03927
               = $character, = $word, = $visible, = $invisible, =
               $link, = $number, = $text:                               03928
                  BEGIN                                                 03929
                  clist (ctcmk, destination.stfile, nofile);            03930
                  dpset(dsprfmt, destination, endfil, destination);
                                                                        03931
                  curmkr ← destination;  curmkr[1] ←
                  destination[1]-1;                                     03932
                  csetctex(&destination, &destination+d2sel,
                  param2);                                              03933
```

```
                clupdt ();                                              03934
                END;                                                    03935
             = $statement:                                              03936
                BEGIN                                                   03937
                clist (ctcfm, destination.stfile, nofile);              03938
                dpset(dsprfmt, destination, endfil, destination);
                                                                        03939
                curmkr ← destination;  curmkr[1] ← 1;                   03940
                csetcsta(destination, param2);                          03941
                clupdt ();                                              03942
                END;                                                    03943
             = $group, = $plex, = $branch:                              03944
                BEGIN                                                   03945
                clist (ctcfm, destination.stfile, nofile);              03946
                dpset(dspallf, destination, endfil, endfil);            03947
                curmkr ← destination;  curmkr[1] ← 1;                   03948
                csetcgro(destination, [&destination+d2sel],
                param2);                                                03949
                clupdt #[]:
                end:
              °mode:                                                    03952
                csetcmod(param2);                                       03953
             ENDCASE err(notyet);                                       03954
          END;                                                          04067
       ENDCASE;                                                         04068
    RETURN(&result);                                                    04069
    END.
                                                                        04070
%insert%                                                                 0815
   (xinsert)    %Execute Insert Command%                                 0816
      PROCEDURE                                                          0817
         %FORMALS%                                                       0818
            (result,          %result record%                           0819
            parsemode,        %parsing, backup, cleanup%                0820
            entity,           %entity type%                             0821
            destination,      %destination pointer%                     0822
            level,  %level adjustment characters%                       0823
            parameter);       %viewspec characters%                     0824
            REF                                                         0825
               result, entity, destination, level, parameter;           0826
      LOCAL                                                            03175
         temp, type, da, cords, x, y, adstr[40];                       03176
      REF da;                                                          04215
      LOCAL STRING                                                      0827
         locstr[500];         % string for date and time %              0828
      LOCAL TEXT POINTER                                                0829
         tp1, tp2;                                                      0830
      %----------------------%                                          0831
      CASE parsemode OF                                                 0832
         = parsing:                                                     0833
            BEGIN                                                       03487
            CASE entity OF                                              03489
               = $link:                                                 03490
                  BEGIN                                                 03491
                  IF parameter.stastr THEN                              03492
                     BEGIN                                              03493
```

```
                IF NOT FIND                                          03494
                   SF(parameter) $(SP/TAB) ('(/'</"--") THEN
                                                                     03495
                      ST parameter ←                                 03496
                         '<, SF(parameter) SE(parameter);            03508
                IF NOT FIND                                          03497
                   SE(parameter) $(SP/TAB) (')/'>) THEN              03498
                      ST parameter ←                                 03499
                         SF(parameter) SE(parameter), '>;            03509
                END;                                                 03500
             lnkprs( &parameter, $adstr);                            03501
             parameter ← adstr[1s];                                  03502
                parameter[1] ← adstr[1s+1];                          03503
             [&parameter+d2sel] ← adstr[1e];                         03504
                [&parameter+d2sel+1] ← adstr[1e+1];                  03505
             END;                                                    03506
          ENDCASE;                                                   03507
       CASE entity OF                                                 0834
          %text/structure entities%                                   0835
             = $character, = $text, = $invisible:                     0836
                BEGIN                                                0837
                clist (ctcmk, destination.stfile, nofile);           0838
                dpset(dsprfmt, destination, endfil,
                destination);                                        0839
                curmkr ← destination[d2sel];                         0840
                   curmkr[1] ← destination[d2sel+1] +
                   parameter[d2sel+1]-parameter[1] - 1;               0841
                cinstex(&destination+d2sel, &parameter,
                &parameter+d2sel, FALSE);                            0842
                clupdt ();                                           0843
                END;                                                 0844
             = $word, = $visible, = $number, = $link:                0845
                BEGIN                                                0846
                clist (ctcmk, destination.stfile, nofile);           0847
                dpset(dsprfmt, destination, endfil,
                destination);                                        0848
                curmkr ← destination[d2sel];                         0849
                   curmkr[1] ← destination[d2sel+1] +
                   parameter[d2sel+1]-parameter[1];                  0850
                cinstex(&destination+d2sel, &parameter,
                &parameter+d2sel, TRUE);                             0851
                clupdt ();                                           0852
                END;                                                 0853
             = $statement:                                           0854
                BEGIN                                                0855
                temp ← 0;                                            03178
                curmkr ← xcmst( &destination, level,
                &parameter, copyflag, 0, $temp);                     03179
                curmkr[1] ← 1;                                        0859
                END;                                                 0861
             = $branch, = $plex, = $group:                           0862
                BEGIN                                                03180
                temp ← 0;                                            03181
                curmkr ← xcmgrp( &destination, level,
                &parameter, copyflag, 0, $temp);                     03182
                curmkr[1] ← 1;                                       03183
```

```
        END;                                                    03184
    = $date, = $time:                                           0868
      BEGIN                                                     0869
      % date (and time) to string; set up pointers %            0870
        *locstr* ← NULL;                                        0871
        getdat( $locstr );                                      0872
        CASE entity OF                                          0873
          =$date:                                               0874
            BEGIN                                               0875
            IF NOT                                              0876
              (FIND SF(*locstr*) $PT (SP ↑tpl))                 0877
                THEN err($"Bad Date From TENEX");
                                                                0878
            ST tpl ← SF(tpl) tpl;                               0879
            END;                                                0880
          ENDCASE;                                              0881
        FIND SF(*locstr*) ↑tpl SE(*locstr*) ↑tp2;               0882
      clist (ctcmk, destination.stfile, nofile);                0883
      dpset(dsprfmt, destination, endfil, destination);
                                                                0884
      curmkr ← destination[d2sel];                              0885
        curmkr[1] ← destination[d2sel+1];                       0886
      cinstex(&destination+d2sel, $tpl, $tp2, TRUE);            0887
      clupdt ();                                                0888
      END;                                                      0889
    = $sendmail: %form%                                         0890
      BEGIN                                                     03281
      *locstr* ←                                                03285
        *sjtitle*, EOL, *sjcomment*, EOL,                       03680
        *sjauthor*, *initsr*, EOL, *sjnumber*, EOL,
                                                                04293
        *sjaction*, EOL, *sjinfo*, EOL, *sjsubcol*,
        EOL,                                                    04294
        *sjkeywords*, EOL, *sjhandling*, EOL,                   04296
        *sjrecording*, EOL, *sjhardcopy*, EOL,                  04297
        *sjrfc*, EOL, *sjobsoletes*, EOL,                       04298
        *sjaccess*, EOL, *sjupdates*, EOL,                      04302
        *sjlink*, EOL, *sjforward*, EOL,                        04299
        *sjmessage*, EOL, *sjbranch*, EOL,                      04300
        *sjplex*, EOL, *sjgroup*, EOL,                          04303
        *sjfile*, EOL, *sjsendit* ;                             04304
      FIND SF(*locstr*) ↑tpl SE(*locstr*) ↑tp2;                 03287
      curmkr ← cinssta(destination, level, $tpl, $tp2);
                                                                03286
      curmkr[1] ← 1;                                            03679
      dpset(dspstrc, curmkr, endfil, curmkr);                   03687
      END;                                                      03288
    = $edge: %of window%                                        04157
      BEGIN                                                     04158
      &da ← destination;                                        04167
      IF da.dafrozen THEN                                       04168
        err($"Cannot split a frozen window");                   04169
      boundry(destination[1] : cords, type);                    04190
        %ignore da returned%                                    04191
      clearda(&da); %erase display image from da%               04174
      clrall(&da, TRUE); %deallocate any strings%               04175
```

```
                IF parameter = $center THEN                          04186
                   BEGIN                                             04198
                   CASE  TYPE OF                                      04192
                      = tbound, = bbound:                            04193
                         BEGIN                                       04187
                         x ← (da.daright-da.daleft)/2;               04170
                         cords.xcord ← (x/da.dahinc)*da.dahinc +
                         da.daleft;                                  04171
                         END;                                        04196
                      = lbound, = rbound:                            04194
                         BEGIN                                       04197
                         y ← (da.dabottom-da.datop)/2;               04172
                         cords.ycord ← (y/da.davinc)*da.davinc +
                         da.datop;                                   04173
                         END;                                        04188
                      ENDCASE;                                       04195
                   END                                               04199
                ELSE                                                 04200
                   BEGIN                                             04203
                   cords.xcord ←
                   ((cords.xcord)/da.dahinc)*da.dahinc;              04201
                   cords.ycord ←
                   ((cords.ycord)/da.davinc)*da.davinc;              04202
                   END;                                              04204
                CASE type OF                                         04176
                   = lbound, = rbound:                               04180
                      IF NOT hsplit(&da, cords, lccords=[]) then

                         err(%"Display area too small");             04179
                   = tbound, = bbound:                               04177
                      IF NOT vsplit(&da, cords, lccords()) THEN
                                                                     04181
                         err(%"Display area too small");             04182
                   ENDCASE err(notyet);                              04183
                dpset(dspallf, endfil, endfil, endfil);              04184
                END;                                                 04165
             ENDCASE err(notyet);                                     0892
          END;                                                       03488
       ENDCASE;                                                       0893
    RETURN(&result);                                                  0894
    END.
                                                                      0895
(xinsstatement) %Execute repeat Insert statement%                     0896
    PROCEDURE                                                         0897
       %FORMALS%                                                      0898
          (result,          %result record%                          0899
          parsemode,        %parsing, backup, cleanup%               0900
          level,  %level adjustment value%                          0901
          source);          %source text for stmt%                   0902
          REF                                                        0903
             result, level, source;                                  0904
    %---------------%                                                0905
    CASE parsemode OF                                                0906
       = parsing:                                                    0907
          BEGIN                                                      0908
          curmkr ← cinssta(curmkr, level, &source, &source+d2sel);
```

```
                                                                    0911
            curmkr[1] ← 1;                                          0912
            dpset(dspstrc, curmkr, endfil, getnxt(curmkr));         0910
            END;                                                    0914
         ENDCASE;                                                   0915
      RETURN( &result );                                            0916
      END.
                                                                    0917
%load%                                                             01093
   (xload) %Execute Load Command%                                  01094
      PROCEDURE                                                    01095
         %FORMALS%                                                 01096
            (result,          %result record%                      01097
            parsemode,        %parsing, backup, cleanup%           01098
            entity,           %type of load%                       01099
            filename);        %name of filestr to be loaded%       01100
         LOCAL da, fileno, stid, pcap, tp;                         01101
         LOCAL STRING filestr[200];                                01104
         REF                                                       01105
            result, filename, entity, da, tp;                      01106
      %----------------------%                                     01107
      CASE parsemode OF                                            01108
         = parsing:                                                01109
            BEGIN                                                  01110
            &da ← cspupdate ← lda();                               01111
            &tp ← &filename+d2sel;                                 01112
            % move file name to local string %                     01113
               CASE lnbfls( &filename, 0, $filestr) OF             03477
                  = lhostn: NULL;                                  03479
                  ENDCASE                                          03480
                     err($"Remote File Manipulations Not Implemented
                     Yet");                                        03481
            CASE entity OF                                         01122
               = $file:                                            01123
                  BEGIN                                            01124
                  IF (fileno ← cloafil($filestr)) THEN             01125
                     BEGIN                                         01126
                     curmkr ← orgstid;                             01127
                     curmkr.stfile ← fileno;                       01128
                     curmkr[1] ← 1;                                01129
                     END;                                          01130
                  END;                                             01131
               = $busy: %file%                                     01132
                  BEGIN                                            01133
                  pcap ← -1;                                       01134
                  IF (pcap ← enablw()) = -1 THEN                   01135
                     err($"This command only available to system
                     personnel");                                  01136
                  IF NOT FIND SF(*filestr*) {'.} THEN              01137
                     *filestr* ← *filestr*, ".NLS";                01138
                  IF (stid ← cloamodfil($filestr)) THEN            01139
                     BEGIN                                         01140
                     curmkr ← stid;                                01141
                     curmkr[1] ← 1;                                01142
                     END;                                          01143
                  disablw(pcap);                                   01144
```

```
                END;                                                    01145
              ENDCASE err(notyet);                                      01146
            cspvs ← da.davspec;                                         01147
            cspvs[1] ← da.davspc2;                                      01148
            dpset(dspyes, curmkr, endfil, endfil);                      01149
            END;                                                        01150
         ENDCASE;                                                       01151
      RETURN(&result);                                                  01152
      END.                                                              01153

%logout%                                                                01154
   (xlogout) %Execute Logout Command%                                   01155
      PROCEDURE                                                         01156
         %FORMALS%                                                      01157
            (result,          %result record%                           01158
            parsemode);       %parsing, backup, cleanup%                01159
            REF                                                         01160
               result;                                                  01161
      %----------------------%                                          01162
      CASE parsemode OF                                                 01163
         = parsing:                                                     01164
            BEGIN                                                       01165
            clogout();                                                  01166
            %clean up this end and logout%                              01167
               IF NOT SKIP !lgout(-1) THEN                              01168
                  err($"Unable to logout");                             01169
            END;                                                        01170
         ENDCASE;                                                       01171
      RETURN(&result);                                                  01172
      END.                                                              01173

%mark%                                                                  01174
   (xmark) %Execute Mark Command%                                       01175
      PROCEDURE                                                         01176
         %FORMALS%                                                      01177
            (result,          %result record%                           01178
            parsemode,        %parsing, backup, cleanup%                01179
            destination,      %char to be marked%                       01180
            mkrname);         %marker name%                             01181
            REF                                                         01182
               result, destination, mkrname;                            01183
      %----------------------%                                          01184
      CASE parsemode OF                                                 01185
         = parsing:                                                     01186
            BEGIN                                                       01187
            cmarcha(&destination, &mkrname, &mkrname+d2sel);            01188
            curmkr ← destination; curmkr[1] ← destination[1];           01189
            END;                                                        01190
         ENDCASE;                                                       01191
      RETURN(&result);                                                  01192
      END.                                                              01193

%merge%                                                                 01194
   (xmerge) %Execute Merge Command%                                     01195
      PROCEDURE                                                         01196
         %FORMALS%                                                      01197
```

```
            (result,          %result record%                          01198
            parsemode,        %parsing, backup, cleanup%               01199
            entity,           %source entity type%                     01200
            source,           %source pointer%                         01201
            destination); %destination pointer%                        01202
            REF                                                        01203
               result, entity, source, destination;                    01204
         %----------------------%                                      01205
         CASE parsemode OF                                             01206
            = parsing:                                                 01207
               CASE entity OF                                          01208
                  = $group, = $plex:                                   01209
                     BEGIN                                             01210
                     cmergro(&destination, &destination+d2sel, &source,
                     &source+d2sel);                                   01211
                     curmkr ← gethed(destination);  curmkr[1] ← 1;     01212
                     dpset(dspstrc, destination, endfil, endfil);      01213
                     END;                                              01214
                  = $branch:                                           01215
                     BEGIN                                             01216
                     IF (destination := getsub(destination)) =
                     destination OR (source := getsub(source)) = source
                     THEN err($"Illegal Merge");                       01217
                     destination[d2sel] ← getail(destination);         01218
                     source[d2sel] ← getail(source);                   01219
                     REPEAT CASE($group);                              01220
                     END;                                              01221
                  ENDCASE err(notyet);                                 01222
            ENDCASE;                                                   01223
         RETURN(&result);                                              01224
         END.                                                          01225

%move%                                                                 01226
   (xmove) %Execute Move Command%                                      01227
      PROCEDURE                                                        01228
         %FORMALS%                                                     01229
            (result,          %result record%                          01230
            parsemode,        %parsing, backup, cleanup%               01231
            sourcentity,      %source entity type%                     01232
            source,                 %source pointer%                   01233
            destentity,       %destination entity type%                01234
            destination,      %destination pointer%                    01235
            level,            %level adjustment string%                01236
            filterflag,       %if TRUE, filtered with viewspecs in vs%
                                                                       01237
            vs);              %viewspec string%                        01238
            REF                                                        01239
               result, sourcentity, source, destentity, destination,
               level, filterflag, vs;                                  01240
         LOCAL                                                         01241
            type, sourceda, destda, da, x, y, r, rhostn, rhost2,
            tlngth, adstr[40];                                         03407
            REF sourceda, destda, da;                                  03734
         LOCAL STRING                                                  03408
            filstr[200], filst2[200];                                  03409
         %----------------------%                                      01245
```

```
CASE parsemode OF                                                   01246
   = parsing:                                                       01247
      BEGIN                                                         03412
      result ← 0;                                                   03703
      CASE sourcentity OF                                           03414
         = $link:                                                   03415
            BEGIN                                                   03416
            IF source.stastr THEN                                   03424
               BEGIN                                                03425
               IF NOT FIND                                          03426
                  SF(source) $(SP/TAB) ('(/'</"--") THEN            03428
                     ST source ← '<, SF(source) SE(source);
                                                                    03427
               IF NOT FIND                                          03429
                  SE(source) $(SP/TAB) (')/'>) THEN                 03430
                     ST source ← SF(source) SE(source), '>;
                                                                    03431
               END;                                                 03432
            lnkprs( &source, $adstr);                               03417
            source ← adstr[ls];                                     03418
               source[1] ← adstr[ls+1];                             03419
            [&source+d2sel] ← adstr[le];                            03420
               [&source+d2sel+1] ← adstr[le+1];                     03421
            END;                                                    03422
         ENDCASE;                                                   03423
      CASE sourcentity OF                                           01248
         %text/structure entities%                                  01249
            = $character, = $invisible, = $text:                    01250
               BEGIN                                                01251
               clist (ctcmk, destination.stfile,
               source.stfile);                                      01252
               dpset(dsprfmt, destination, source, endfil);
                                                                    01253
               curmkr ← destination[d2sel];                         01254
                  curmkr[1] ← destination[d2sel+1] +
                  source[d2sel+1]-source[1]-1;                      01255
               IF NOT source.stastr THEN                            01256
                  cmovtex(&destination+d2sel, &source,
                  &source+d2sel, FALSE)                             01257
               ELSE                                                 01258
                  cinstex(&destination+d2sel, &source,
                  &source+d2sel, FALSE);                            01259
               clupdt ();                                           01260
               END;                                                 01261
            = $word, = $visible, = $number, = $link:                01262
               BEGIN                                                01263
               clist (ctcmk, destination.stfile,
               source.stfile);                                      01264
               dpset(dsprfmt, destination, source, endfil);
                                                                    01265
               curmkr ← destination[d2sel];                         01266
                  curmkr[1] ← destination[d2sel+1] +
                  source[d2sel+1]-source[1];                        01267
               IF NOT source.stastr THEN                            01268
                  cmovtex(&destination+d2sel, &source,
                  &source+d2sel, TRUE)                              01269
```

```
                ELSE                                                    01270
                   cinstex(&destination+d2sel, &source,
                   &source+d2sel, TRUE);                                01271
                clupdt ();                                              01272
                END;                                                    01273
             = $statement:                                              01274
                BEGIN                                                   01275
                curmkr ← xcmst(&destination, level, &source,
                moveflag, filterflag, &vs);                             01276
                curmkr[1] ← 1;                                          01277
                END;                                                    01278
             = $group, = $plex, = $branch:                              01279
                BEGIN                                                   01280
                curmkr ← xcmgrp(&destination, level, &source,
                moveflag, filterflag, &vs);                             01281
                curmkr[1] ← 1;                                          01282
                END;                                                    01283
          = $file:                                                      01284
            BEGIN                                                       01285
            % get and initialize message string %                       03716
               result ← getstring( 3000, $dspblk);                      03707
               *[result]* ← "Moved Files Are:
               ";                                                       03708
               tlngth ← [result].L;                                     03709
            % parse source file name %                                  01286
               rhostn ← lnbfls( &source, 0, $filstr);                   03475
            % parse destination file name %                             01290
               rhost2 ← lnbfls( &destination, 0, $filst2);
                                                                        03476
            cmovfil(rhostn, $filstr, rhost2, $filst2, result);
                                                                        01294
            % tell the user what we did %                              03717
               IF ( [result].L > tlngth ) THEN                          03712
                  fbctl( typecalit, result)                             03713
               ELSE fbctl( typecalit, $"No Files Moved");               03714
            END;                                                        01295
          = $edge:                                                      01296
            BEGIN                                                       01297
            &sourceda ← dsparea(boundry(source[1] : source[1],
            type));                                                     01300
               %get boundary nearest cursor%                            01299
            IF sourceda.dafrozen THEN                                   01301
               err($"Can't move boundary of a frozen window");
                                                                        01302
            &destda ← destination;                                      01303
            IF destda.dafrozen THEN                                     01320
               err($"Can't move boundary of a frozen window");
                                                                        01321
            IF destentity = $center THEN                                04205
               BEGIN                                                    04206
               &da ← findda(destination[1]);                            04207
               CASE type OF                                             04208
                  = lbound, = rbound:                                   04209
                     destination[1].xcord ←
                     (da.daright-da.daleft)/2;                          04214
                  = tbound, = bbound:                                   04210
```

```
                        destination[1].ycord ←
                        (da.dabottom-da.datop)/2;                    04213
                     ENDCASE;                                        04211
                  END;                                               04212
               x ← destination[1].xcord - destda.daleft;             01304
               IF destination[1].xcord < (destda.daright -
               destda.dahinc/2) THEN                                 01305
                  BEGIN                                              01306
                  DIV x / destda.dahinc, x, r;                       01307
                  IF r >= (destda.dahinc/2) THEN BUMP x;             01308
                  x ← x * destda.dahinc;                             01309
                  destination[1].xcord ← x + destda.daleft;          01310
                  END;                                               01311
               y ← destination[1].ycord - destda.datop;              01312
               IF destination[1?).ycord < (destda.dabottom -
               destda.davinc/2) THEN                                 01313
                  BEGIN                                              01314
                  DIV y / destda.davinc, y, r;                       01315
                  IF r >= (destda.davinc/2) THEN BUMP y;             01316
                  y ← y * destda.davinc;                             01317
                  destination[1].ycord ← y + destda.datop;           01318
                  END;                                               01319
               clearda(0); %erase entire text area%                  01322
               clrall(0, TRUE); %erase all line seg ref tables%
                                                                     01323
               movbndry(&sourceda, source[1], destination[1],
               type);                                                01324
               dpset(dspallf, endfil, endfil, endfil);               01325
               END;                                                  01326
            ENDCASE err(notyet);                                     01327
         END;                                                        03413
      = backup, = cleanup:                                           03704
         IF result THEN freestring(result, $dspblk);                 03705
      ENDCASE;                                                       01326
   RETURN(&result);                                                  01329
   END.

                                                                     01330
%output%                                                             01331
   (xoutsapf) %set oqapfg for output commands%                       01332
      PROCEDURE                                                      01333
         %FORMALS%                                                   01334
            (result,          %result record%                        01335
            parsemode,        %parsing, backup, cleanup%             01336
            parameter);       %value for oqapfg%                     01339
         REF                                                         01344
            result, parameter;                                       01345
      %----------------------%                                       01346
      CASE parsemode OF                                              01347
         = parsing:                                                  01348
            oqapfg ← parameter;                                      03269
         ENDCASE oqapfg ← FALSE;                                     01411
      RETURN(&result);                                               01412
      END.

                                                                     01413
   (xoutsnhf) %set oqnhfg for output commands%                       04326
      PROCEDURE                                                      04327
```

```
        %FORMALS%                                                       04328
            (result,        %result record%                             04329
            parsemode,      %parsing, backup, cleanup%                  04330
            parameter);     %value for oqnhfg%                          04331
        REF                                                             04332
            result, parameter;                                          04333
        %----------------------%                                        04334
        CASE parsemode OF                                               04335
            = parsing:                                                  04336
                oqnhfg ← parameter;                                     04337
            ENDCASE oqnhfg ← FALSE;                                     04338
        RETURN(&result);                                                04339
        END.                                                            04340

    (xout1) %Execute Output (quickprint, journal, printer, com)
    Command%                                                            03187
        PROCEDURE                                                       03188
            %FORMALS%                                                   03189
                (result,        %result record%                         03190
                parsemode,      %parsing, backup, cleanup%              03191
                entity,         %entity type%                           03192
                filename,       %file name pointer%                     03193
                parameter,      %viewspec characters%-                  03194
                tstpar);        %$test or NULL%                         04152
            LOCAL tp;                                                   03195
            LOCAL STRING locstr[200];                                   03198
            REF                                                         03199
                result, entity, filename, parameter, tp, tstpar;        03200
            %----------------------%                                    03201
            CASE parsemode OF                                           03202
                = parsing:                                              03203
                    BEGIN                                               03204
                    % decode number of copies string if necessary %     03205
                        IF parameter > 1 THEN                           03206
                            BEGIN                                       03207
                            &tp ← &parameter + d2sel;                   03208
                            *locstr* ←  parameter tp;                   03209
                            ndr( &parameter, &parameter, &parameter );  03855
                            parameter ← VALUE($locstr);                 03210
                            END;                                        03211
                    % get output file name to locstr, use dfltname to
                    construct the name if none was specified %          03212
                        IF filename                                     03213
                            THEN % use name supplied by user %          03214
                                CASE lnbfls( &filename, 0, $locstr) OF  03402
                                    = lhostn: NULL;                     03404
                                    ENDCASE                             03405
                                        err($"Remote File Manipulations Not
                                        Implemented Yet")               03406
                            ELSE % construct default name %             03225
                                dfltname( $locstr, entity, tstpar );    03226
                    CASE entity OF                                      03227
                        = $quickprint:                                  03228
                            BEGIN                                       03229
                            IF NOT FIND SF(*locstr*) /'./ THEN          03230
                                *locstr* ← *locstr*, '., STRING(parameter);
```

```
                                                                    03231
                    %make ext be number of copies%                  03232
                coutqui($locstr, lda());                            03233
                END;                                                03234
            = $journal: %submission form%                           03235
                BEGIN                                               03236
                IF NOT FIND SF(*locstr*) ['.] THEN                  03237
                   *locstr* ← *locstr*, '., STRING(parameter);
                                                                    03238
                    %make ext be number of copies%                  03239
                coutjouqui($locstr, lda());                         03240
                END;                                                03241
            = $printer:                                             03242
                BEGIN                                               03243
                IF NOT FIND SF(*locstr*) ['.] THEN                  03244
                   *locstr* ← *locstr*  '., STRING(parameter);
                                                                    03245
                    %make ext be number of copies%                  03246
                coutproc($locstr, lda(), oppradv, 0);               03247
                END;                                                03248
            = $com:                 %Computer output to Microfilm%
                                                                    03249
                BEGIN                                               03250
                IF NOT FIND SF(*locstr*) ['.] THEN                  03251
                   *locstr* ← *locstr*, '., STRING(parameter);
                                                                    03252
                    %make ext be number of copies%                  03253
                coutproc($locstr, lda(),  IF tstpar = $test THEN
                opxpdv ELSE opcmdv, 0);                             03254
                END;                                                03255
            = $assembler:                                           03256
                BEGIN                                               03257
                coutassfil($locstr, lda(), parameter);              03258
                END;                                                03259
            = $sequential:                                          03260
                BEGIN                                               03261
                coutseqfil($locstr, lda(), parameter);              03262
                END;                                                03263
            ENDCASE err(notyet);                                    03264
        END;                                                        03265
    ENDCASE;                                                        03266
RETURN(&result);                                                    03267
END.

                                                                    03268
(xout2) %Output (Terminal, Remote printer) Command%                 01414
  PROCEDURE                                                         01415
    %FORMALS%                                                       01416
      (result,          %result record%                             01417
      parsemode,        %parsing, backup, cleanup%                  01418
      entity,           %entity type%                               01419
      tip,              %tip name or filename if an output
      terminal file%                                                01420
      tipport,          %tip port%                                  01421
      formfeed,         %TRUE: send FF, FALSE: see simff%           01422
      simff,  %TRUE: simulate FF%                                   01423
      waitpb);          %TRUE: wait at page break%                  01424
```

```
    LOCAL opflags, devtype, tp;                                    01425
    LOCAL STRING tipstr[10], trmstr[10], outfile[30];              01426
    REF result, entity, tip, tipport, formfeed, simff, waitpb,
    tp;                                                            01427
%----------------------%                                          01428
CASE parsemode OF                                                  01429
   = parsing:                                                      01430
      BEGIN                                                        01431
      % setup flags record (to be passed to OP) %                  01432
      opflags ← 0;                                                 01433
      CASE formfeed OF                                             01434
         = 1:                                                      01435
            BEGIN                                                  01436
            opflags.opform ← TRUE;                                 01437
            opflags.opsimff ← FALSE;                               01438
            END;                                                   01439
         = 2, = 0:                                                 01440
            BEGIN                                                  01441
            opflags.opform ← FALSE;                                01442
            CASE simff OF                                          01443
               = 1:    opflags.opsimff ← TRUE;                     01444
               = 2, = 0:       opflags.opsimff ← FALSE;            01445
               ENDCASE err($"invalid response");                   01446
            END;                                                   01447
         ENDCASE err($"invalid response");                         01448
      CASE waitpb OF                                               01449
         = 1:          opflags.opwtpb ← TRUE;                      01450
         = 2, = 0;     opflags.opwtpb ← FALSE;                     01451
         ENDCASE err($"invalid response");                         01452
      % construct file name based of device %                      01453
      CASE entity OF                                               01454
         = $terminal:                                              01455
            BEGIN                                                  01456
            devtype ← optydv;                                      01457
            IF NOT tip THEN *outfile* ← "TTY:"                     04639
            ELSE                                                   04640
               CASE lnpfis( &tip, 0, $outfile) OF                  04641
                  = lhostn: NULL;                                  04642
                  ENDCASE                                          04643
                     err($"Remote File Manipulations Not
                     Implemented Yet");                            04644
            END;                                                   01459
         = $remote: %printer/terminal%                             01460
            BEGIN                                                  01461
            &tp ← &tip + d2sel;                                    01462
            *tipstr* ← tip tp;                                     01463
            &tp ← &tipport + d2sel;                                01464
            *trmstr* ← tipport tp;                                 01465
            devtype ← oprmdv;                                      01466
            *outfile* ← "NET:0,",                                  01467
               STRING(VALUE($tipstr), 8), '-,
                STRING((VALUE($trmstr)* 65536 + 2), 8);            01468
            END;                                                   01469
         ENDCASE err(notyet);                                      01470
      coutproc($outfile, lda(), devtype, opflags);                 01471
      END;                                                         01472
```

```
      ENDCASE;                                                          01473
   RETURN(&result);                                                     01474
   END.                                                                 01475

(dfltname) % construct default output file name %                       01476
   PROCEDURE(                                                           01477
   % FORMAL ARGUMENTS %                                                 01478
      str,        % ptr to result astring %                            01479
      type,    % $quickprint, $com, etc. %                             04476
      tstpar); % $test OR NULL %                                        04481
   LOCAL TEXT POINTER tp1, tp2;                                         01480
   REF % VARIABLES %                                                    01481
      str;                                                              01482
   % ------------------- %                                             01483
   %put file name into a string%                                        01484
      *str* ← NULL;                                                     01485
      filnam (fida()).dacsp.stfile, &str);                              01486
   % check it %                                                         01487
      IF NOT (FIND SF(*str*) ',/ $SP ↑tp1 './ < CH ↑tp2) THEN
                                                                        01488
         err ($"bad file name");                                        01489
   % edit string, putting printer directory and user's initials
   into it %                                                            01490
      *str* ← '(, *initsr*, '), tp1 tp2;                                01491
      IF type = $com AND NOT tstpar THEN                                04478
         *str* ← "<COM>", *str*, ".COM"                                 04479
      ELSE *str* ← '<,  *prtdir*, '>, *str*;                            04480
   RETURN;                                                              01492
   END.                                                                 01493

%playback%                                                              01494
   (xplayback) %Execute Playback Command%                               01495
      PROCEDURE                                                         01496
         %FORMALS%                                                      01497
            (result,          %result record%                         01498
            parsemode,        %parsing, backup, cleanup%              01499
            filename,         %name of file to be played%             01500
            symrt);                    % flag to simulate recorded
            timing%                                                     04616
            REF                                                         01501
               result, filename, symrt;                                 01502
      LOCAL                                                             01503
         rhostn;                                                        01504
      LOCAL TEXT POINTER f1, f2;                                        03645
      LOCAL STRING                                                      01505
         filstr[200];                                                   01506
      %----------------------%                                         01507
      CASE parsemode OF                                                 01508
         = parsing:                                                     01509
            BEGIN                                                       01510
            % move file name to local string %                          01511
               rhostn ← lnbfls( &filename, 0, $filstr);                 03401
            % setup text pointers to start and end of string %         01520
               FIND SF(*filstr*) ↑f1 SE(*filstr*) ↑f2;                  01521
            % set global flag symtflg %                                04617
               symtflg ← symrt;                                         04618
```

```
            xrecplasup( FALSE, $f1, $f2, rhostn);                    01522
            END;                                                     01523
         ENDCASE;                                                    01524
      RETURN(&result);                                               01525
      END.
                                                                     01526
%print%                                                              01527
   (xprint) %Execute TNLS Print Command%                             01528
      PROCEDURE                                                      01529
         %FORMALS%                                                   01530
            (result,          %result record%                        01531
            parsemode,        %parsing, backup, cleanup%             01532
            entity,           %entity type%                          01533
            destination,      %destination pointer%                  01534
            vs);              %viewspec record%                      01535
            REF                                                      01536
               result, entity, destination, vs;                      01537
         LOCAL da;  REF da;                                          01538
         LOCAL vssav1, vssav2, cspsav;      %Save current VS and SP%
                                                                     04306
      %-----------------------%                                     01539
      CASE parsemode OF                                              01540
         = parsing:                                                  01541
            BEGIN                                                    01542
            &da ← lda();                                             01543
            CASE entity OF                                           01544
               %structure entities%                                  01545
                  = $statement:                                      01546
                     BEGIN                                           01547
                     curmkr ← destination;  curmkr[1] ←
                     destination[1];                                 01548
                     cprista(destination, &vs, &da);                 01549
                     cspvs ← da.davspec;  cspvs[1] ← da.davspc2;
                                                                     03790
                     END;                                            01550
                  = $group, = $branch, = $plex:                      01551
                     BEGIN                                           01552
                     curmkr ← destination;  curmkr[1] ←
                     destination[1];                                 01553
                     cprigro(destination, [&destination+d2sel], &vs,
                     &da);                                           01554
                     cspvs ← da.davspec;  cspvs[1] ← da.davspc2;
                                                                     03791
                     END;                                            01555
                  = $rest, = $file:                                  01556
                     BEGIN                                           04307
                     cspsav ← da.dacsp;        %save current SP%     04315
                     vssav1 ← da.davspec;      %save current VS%     04308
                     vssav2 ← da.davspc2;                            04309
                     %da.davspec ← stdvsp;%    %use standard VS%     04310
                     %da.davspc2 ← stdvsp[1];%                       04311
                     IF entity = $file THEN da.dacsp.stpsid ← orgstid;
                                                                     04316
                     ON SIGNAL ELSE                                  04317
                        BEGIN                                        04318
                        da.dacsp ← cspsav;                           04319
```

```
                    da.davspec ← vssav1;                                    04320
                    da.davspc2 ← vssav2;                                    04321
                    RETURN(&result);                                        04322
                    END;                                                    04323
                 cprires(da.dacsp, &da);                                    01557
                 ON SIGNAL ELSE;                                            04324
                 da.dacsp ← cspsav;                                         04325
                 da.davspec ← vssav1;      %restore current VS%
                                                                            04312
                 da.davspc2 ← vssav2;                                       04313
                 END;                                                       04314
              = $journal:                                                   03788
                 cprijou(da.dacsp, &da);                                    03789
              ENDCASE err(notyet);                                          01558
           END;                                                             01559
        ENDCASE;                                                            01560
     RETURN(&result);                                                       01561
     END.
                                                                            01562
%process%                                                                   01694
   (xprocess) %Execute Process Command%                                     01695
      PROCEDURE                                                             01696
         %FORMALS%                                                          01697
            (result,          %result record%                               01698
            parsemode,        %parsing, backup, cleanup%                    01699
            destentity,       % destination entity type %                   01701
            destination);% destination record %                             01702
            REF                                                             01703
               result, entity, destentity, destination;                     01704
         LOCAL TEXT POINTER endtptr; % points to last char %                01705
      %----------------------%                                              01706
      CASE parsemode OF                                                     01707
         = parsing:                                                         01708
            BEGIN                                                           01711
            endtptr ← destination[d2sel];                                   01712
            endtptr[1] ← destination[d2sel+1];                              01713
            CASE destentity OF                                              01714
               = $statement:                                                01715
                  NULL;                                                     01716
               = $branch, = $group, = $plex:                                01717
                  BEGIN                                                     01718
                  endtptr ← getend(endtptr);                                01719
                  FIND SE(endtptr) ↑endtptr;                                01720
                  END;                                                      01721
            ENDCASE err(notyet);                                            01722
            auxstartup( &destination, $endtptr );                           01723
            END;                                                            01724
         ENDCASE;                                                           01726
      RETURN(&result);                                                      01727
      END.
                                                                            01728
                                                                            04305
%record%                                                                    01729
   (xstart) %Execute Start Record Command%                                  01730
      PROCEDURE                                                             01731
         %FORMALS%                                                          01732
```

```
            (result,          %result record%                    01733
            parsemode,        %parsing, backup, cleanup%         01734
            filename);        %name of file to be archived%      01735
            REF                                                  01736
               result, filename;                                 01737
      LOCAL rhostn;                                              03395
      LOCAL TEXT POINTER f1, f2;                                 03646
      LOCAL STRING                                               01740
         filstr[200];                                            01741
      %----------------------%                                   01742
      CASE parsemode OF                                          01743
         = parsing:                                              01744
            BEGIN                                                01745
            % move file name to local string %                   01746
               rhostn ← lnbfls( &filename, 0, $filstr);          03390
            % point to start and end of the string %            01755
               FIND SF(*filstr*) ↑f1 SE(*filstr*) ↑f2;           01756
            xrecplasup( TRUE, $f1, $f2, rhostn);                 01757
            END;                                                 01758
         ENDCASE;                                                01759
      RETURN(&result);                                           01760
      END.                                                       01761

%renumber%                                                       01794
   (xrenumber) %Execute Renumber Command%                        01795
      PROCEDURE                                                  01796
         %FORMALS%                                               01797
            (result,          %result record%                    01798
            parsemode);       %parsing, backup, cleanup%         01799
            REF                                                  01800
               result;                                           01801
      %----------------------%                                   01802
      CASE parsemode OF                                          01803
         = parsing:                                              01804
            BEGIN                                                01805
            crensidfil(lcfile());                                01806
            dpset(dspyes, [lda()].dacsp, endfil, endfil);        01807
            END;                                                 01808
         ENDCASE;                                                01809
      RETURN(&result);                                           01810
      END.                                                       01811

%replace%                                                        01812
   (xreplace) %Execute Replace Command%                          04343
      PROCEDURE                                                  04344
         %FORMALS%                                               04345
            (result,          %result record%                    04346
            parsemode,        %parsing, backup, cleanup%         04347
            destentity,       %destination entity type%          04348
            destination,      %destination pointer%              04349
            sourcentity,      %source entity type%               04350
            source);          %source pointer%                   04351
            REF                                                  04352
               result, sourcentity, source, destentity, destination;
                                                                 04353
         LOCAL delt;                                             04354
```

```
    LOCAL TEXT POINTER tp1, tp2;                                    04355
    LOCAL STRING temp[40];                                          04356
    LOCAL adstr[40];                                                04357
  %----------------------%                                          04358
  CASE parsemode OF                                                 04359
    = parsing:                                                      04360
      BEGIN                                                         04361
      CASE sourcentity OF                                           04362
        = $link:                                                    04363
          BEGIN                                                     04364
          IF source.stastr THEN                                     04365
            BEGIN                                                   04366
            IF NOT FIND                                             04367
              SF(source) $(SP/TAB) ('(/'</"--") THEN                04368
                ST source ← '<, SF(source) SE(source);
                                                                    04369
            IF NOT FIND                                             04370
              SE(source) $(SP/TAB) (')/'>) THEN                     04371
                ST source ← SF(source) SE(source), '>;
                                                                    04372
            END;                                                    04373
          lnkprs( &source, $adstr);                                 04374
          source ← adstr[ls];                                       04375
            source[1] ← adstr[ls+1];                                04376
          [&source+d2sel] ← adstr[le];                              04377
            [&source+d2sel+1] ← adstr[le+1];                        04378
          END;                                                      04379
        ENDCASE;                                                    04380
      CASE destentity OF                                            04381
        = $link:                                                    04382
          BEGIN                                                     04383
          IF destination.stastr THEN                                04384
            BEGIN                                                   04385
            IF NOT FIND                                             04386
              SF(destination) $(SP/TAB) ('(/'</"--") THEN
                                                                    04387
                ST destination ←                                    04388
                  '<, SF(destination) SE(destination);
                                                                    04389
            IF NOT FIND                                             04390
              SE(destination) $(SP/TAB) (')/'>) THEN                04391
                ST destination ←                                    04392
                  SF(destination) SE(destination), '>;
                                                                    04393
            END;                                                    04394
          lnkprs( &destination, $adstr);                            04395
          destination ← adstr[ls];                                  04396
            destination[1] ← adstr[ls+1];                           04397
          [&destination+d2sel] ← adstr[le];                         04398
            [&destination+d2sel+1] ← adstr[le+1];                   04399
          END;                                                      04400
        = $number:                                                  04401
          BEGIN                                                     04402
          delt ← destination[d2sel+1] - destination[1] -
          source[d2sel+1] +source[1];                               04403
          CASE delt OF                                              04404
```

```
                < 0:                                                    04405
                   BEGIN                                                04406
                   LOOP                                                 04407
                      IF (delt := delt+1) >= 0 OR                       04408
                      (NOT FIND destination < SP SP
                      ↑destination ←destination) THEN EXIT
                      LOOP;                                             04409
                   END;                                                 04410
                > 0:                                                    04411
                   BEGIN                                                04412
                   tp1 ← source;                                        04413
                   tp1[1] ← source[1];                                  04414
                   tp2 ← source[d2sel];                                 04415
                   tp2[1] ← source[d2sel+1];                            04416
                   *temp* ← NULL;                                       04417
                   UNTIL (delt ← delt-1) < 0 DO *temp* ←
                   *temp*, SP;                                          04418
                   *temp* ← *temp*, tp1 tp2;                            04419
                   FIND SF(*temp*) ↑tp1 SE(*temp*) ↑tp2;                04420
                   source ← tp1;                                        04421
                   source[1] ← tp1[1];                                  04422
                   source[d2sel] ← tp2;                                 04423
                   source[d2sel+1] ← tp2[1];                            04424
                   END;                                                 04425
                ENDCASE;                                                04426
             END;                                                       04427
          ENDCASE;                                                      04428
       CASE sourcentity OF                                              04429
          %text/structure entities%                                     04430
             = $character, = $word, = $visible, = $invisible, =
             $link, = $number, = $text, = $statement:                   04431
                BEGIN                                                   04432
                clist (ctcmk, destination.stfile,
                source.stfile);                                         04433
                dpset(dsprfmt, destination, endfil, endfil);
                                                                        04434
                curmkr ← destination;  curmkr[1] ←
                destination[1]+source[d2sel+1]-source[1]-1;
                                                                        04435
                creptex(&destination, &destination+d2sel,
                &source, &source+d2sel);                                04436
                clupdt ();                                              04437
                END;                                                    04438
             = $branch, = $plex, = $group:                              04439
                BEGIN                                                   04440
                clist (ctlcfm, destination.stfile,
                source.stfile);                                         04441
                dpset(dsprfst, destination, endfil,
                dpstp(destination));                                    04442
                curmkr ← crepgro(NOT source.stastr,
                &destination, &destination+d2sel, &source,
                &source+d2sel);                                         04443
                clupdt ();                                              04444
                curmkr[1] ← 1;                                          04445
                END;                                                    04446
          ENDCASE err(notyet);                                          04447
```

```
            END;                                                    04448
         ENDCASE;                                                   04449
      RETURN(&result);                                              04450
      END.
                                                                    04451
%reset%                                                             01849
   (xreset) %Execute Reset Command%                                 01850
      PROCEDURE                                                     01851
         %FORMALS%                                                  01852
            (result,          %result record%                       01853
            parsemode,        %parsing, backup, cleanup%            01854
            entity,           %entity type%                         01855
            destentity,       %level adjustment characters%         01856
            destination); %destination pointer%                     01857
            REF                                                     01858
               result, entity, destentity, destination ;            01859
         LOCAL da, save, rhostn;  REF da;                           01860
      LOCAL STRING filstring[200];                                  03361
      %----------------------%                                      01863
      CASE parsemode OF                                             01864
         = parsing:                                                 01865
            BEGIN                                                   01866
            dpset(dspno, endfil, endfil, endfil);                   01867
            CASE entity OF                                          01868
               = $archive: %request for file%                       01869
                  BEGIN                                             01870
                  rhostn←lnbfls(&destination, 0, $filstring);       03362
                  *lit* ← NULL;                                     04142
                  carcfil( rhostn, $filstring, 0, $lit);            04143
                  IF lit.L THEN                                     04144
                     *lit* ← "The archive status of the following
                     files has been changed:", EOL, *lit*           04145
                  ELSE                                              04146
                     *lit*  ← "No files' archive status changed";
                                                                    04147
                  fbctl( typecalit, $lit);                          04148
                  END;                                              04149
               = $case: %mode%                                      01876
                  crescasmod();                                     01877
               = $character: %size for window%                      01878
                  setcharsize(lda(), tacsize);                      01879
               = $content: %Content Analysis%                       01882
                  BEGIN                                             01883
                  &da ← lda();                                      01884
                  da.dacacode ← 0;                                  01885
                  da.davspec.vscapf ← FALSE;                        01886
                  END;                                              01887
               = $link: %default for file%                          01888
                  creslindef(lcfile());                             01889
               = $name: %delimiters in destentity%                  01890
                  BEGIN                                             01891
                  CASE destentity OF                                01892
                     = $statement:                                  01893
                        cresnsta(destination, lda());               01894
                     = $group, = $branch, = $plex:                  01895
                        cresngrc(destination, [&destination+d2sel],
```

```
                        lda());                                          01896
                     ENDCASE err(notyet);                                01897
                  END;                                                   01898
               = $temporary: %modifications to file%                     01910
                  crestemmod(lcfile(), FALSE);                           01911
               = $tty: %window%                                          01912
                  BEGIN                                                  01913
                  clearda(0);                                            04491
                  clrall(0, TRUE);                                       04492
                  %delete current one%                                   03752
                     IF ttysim := 0 THEN dealocda(ttyda);                03753
                  %restore old text display%                             01928
                     IF (&da ← ttyda) NOT= &msgda AND da.daexis THEN
                                                                         03756
                        BEGIN                                            03757
                        da.daseq ← FALSE;                                03758
                        da.dasuppress ← FALSE;                           03761
                        da.dahandle ← alocda(&da);                       03759
                        END;                                             03760
                     dpset(dspallf, endfil, endfil, endfil);             01936
                  %re-allocate system default tty-sim area%              01918
                     save ← linkcnsl := 0;                               01919
                     ttysim ← alocda(&msgda);                            01920
                     linkcnsl ← save;                                    01921
                     ttyda ← &msgda;                                     03754
                     ttywindow(&msgda);                                  03755
                  defttysim ← TRUE;                                      01937
                  END;                                                   01938
               = $viewspecs:                                             01939
                  BEGIN                                                  01940
                  cspupdate ← &da ← lda();                               01941
                  cspvs ← stdvsp;                                        01942
                  cspvs[1] ← stdvsp[1];                                  01943
                  curmkr ← da.dacsp;  curmkr[1] ← da.daccnt;             04495
                  dpset(dspyes, da.dacsp, endfil, endfil);               04494
                  END;                                                   01944
               ENDCASE err(notyet);                                      01945
            END;                                                         01946
         ENDCASE;                                                        01947
      RETURN(&result);                                                   01948
      END.
                                                                         01949
%retrieve%                                                               01950
   (xretrieve) %Execute Retrieve Command%                                01951
      PROCEDURE                                                          01952
         %FORMALS%                                                       01953
            (result,          %result record%                           01954
            parsemode,        %parsing, backup, cleanup%                01955
            filename);        %filename pointer%                        01956
            REF                                                          01957
               result, filename;                                         01958
      LOCAL rhostn;                                                      03360
      LOCAL STRING filstring[200];                                       03354
      %----------------------%                                          01961
      CASE parsemode OF                                                  01962
         = parsing:                                                      01963
```

```
            BEGIN                                                   01964
            rhostn ← lnbfls( &filename, 0, $filstring);             03355
            cretarcfil(rhostn, $filstring);                         01968
            END;                                                    01969
         ENDCASE;                                                   01970
      RETURN(&result);                                              01971
      END.
                                                                    01972
%set%                                                               01973
   (xset) %Execute set Command%                                     01974
      PROCEDURE                                                     01975
         %FORMALS%                                                  01976
            (result,          %result record%                       01977
            parsemode,        %parsing, backup, cleanup%            01978
            entity,           %entity type%                         01979
            param1,           %parameter one%                       01980
            param2,           %parameter two%                       01981
            param3,           %parameter three%                     01982
            destination); %destination pointer%                     01983
            REF                                                     01984
               result, entity, param1, param2, param3, destination;
                                                                    01985
         LOCAL mask, prot, count, i, rhostn, csize, ninc  vinc, da,
         end1, save, tp2, stid, adstr[40];                          01986
         LOCAL STRING sizestring[10], filstr[200];                  01987
         REF da, tp2;                                               01988
      %----------------------%                                      01989
      CASE parsemode OF                                             01990
         = parsing:                                                 01991
            BEGIN                                                   01992
            dpset(dspno, endfil, endfil, endfil);                   01993
            CASE entity OF                                          01994
               = $case: %mode%                                      01995
                  BEGIN                                             01996
                  param2 ← CASE param2 OF                           01997
                     =0: xsmode; % none specified %                 01998
                     =$first: iupcase;                              01999
                     =$upper: upcase;                               02000
                     =$lower: lowcase;                              02001
                     ENDCASE err( notyet );                         02002
                  CASE param1 OF                                    03510
                     = $link:                                       03511
                        BEGIN                                       03512
                        lnkprs( &destination, $adstr);              03522
                        destination ← adstr[ls];                    03523
                           destination[1] ← adstr[ls+1];            03524
                        [&destination+d2sel] ← adstr[le];           03525
                           [&destination+d2sel+1] ← adstr[le+1];
                                                                    03526
                        END;                                        03527
                     ENDCASE;                                       03528
                  CASE param1 OF                                    02003
                     = $character, = $word, = $visible, =
                     $invisible, = $link, = $number, = $text:       02004
                        BEGIN                                       02005
                        clist (ctcmk, destination.stfile, nofile);
```

```
                                                                    02006
                dpset(dsprfmt, destination, endfil,
                destination);                                       02007
                curmkr ← destination;  curmkr[1] ←
                destination[1]-1;                                   02008
                csetctex(&destination, &destination+d2sel,
                param2);                                            02009
                clupdt ();                                          02010
                END;                                                02011
             = $statement:                                          02012
                BEGIN                                               02013
                clist (ctcfm, destination.stfile, nofile);
                                                                    02014
                dpset(dsprfmt, destination, endfil,
                destination);                                       02015
                curmkr ← destination;  curmkr[1] ← 1;               02016
                csetcsta(destination, param2);                      02017
                clupdt ();                                          02018
                END;                                                02019
             = $group, = $plex, = $branch:                          02020
                BEGIN                                               02021
                clist (ctcfm, destination.stfile, nofile);
                                                                    02022
                dpset(dspallf, destination, endfil, endfil);
                                                                    02023
                curmkr ← destination;  curmkr[1] ← 1;               02024
                csetcgrc(destination, (&destination+d2sel),
                param2);                                            02025
                clupdt ();                                          02026
                END;                                                02027
             = $mode:                                               02028
                csetcmod(param2);                                   02029
             ENDCASE err(notyet);                                   02030
          END;                                                      02031
       = $character: %size for window%                              02032
          BEGIN                                                     02033
          % convert number string to value %                        02036
             &tp2 ← &param1 + d2sel;                                02037
             *sizestring* ← param1 tp2;                             02038
             csize ← VALUE($sizestring);                            02039
          setcharsize(1da(), csize);          % go fix up da %
                                                                    03155
          END;                                                      02085
       = $external:  % names link file address %                    03810
          BEGIN                                                     03812
          stid ← orgstid;                                           03831
          stid.stfile ← (1da()).dacsp.stfile;                       03832
          dpset(dsprfmt, stid, endfil, endfil);                     03830
          IF param1.stastr THEN                                     03813
             BEGIN                                                  03814
             IF NOT FIND                                            03815
                SF(param1) $(SP/TAB) ('(/'</"--") THEN              03816
                   ST param1 ←                                      03817
                      '<, SF(param1) SE(param1);                    03818
             IF NOT FIND                                            03819
                SE(param1) $(SP/TAB) (')/'>) THEN                   03820
```

```
                ST param1 ←                                          03821
                   SF(param1) SE(param1), '>;                        03822
            END;                                                     03823
         lnkprs( &param1, $adstr);                                   03824
         csetextname( stid.stfile, $adstr );                         03811
         END;                                                        03829
      = $content: %Content Analysis%                                 02091
         CASE param1 OF                                              02092
            = $to: %pattern%                                         02093
               BEGIN                                                 02094
               cspupdate ← lda();                                    02095
               cspcacode ← cpconan(&param2, &da);                    02096
               END;                                                  02098
            = 1: %on%                                                02099
               BEGIN                                                 03772
               cspupdate ← lda();                                    03770
               cspvs.vscapf ← TRUE;                                  02100
               END;                                                  03773
            = 2: %off%                                               02101
               BEGIN                                                 03774
               cspupdate ← lda();                                    03771
               cspvs.vscapf ← FALSE;                                 02102
               END;                                                  03775
            ENDCASE err(notyet);                                     02103
      = $link: %default for file%                                    02104
         csetlindef(lcfile(), &param2, &param2+d2sel);               02105
      = $name: %delimiters in destentity%                            02106
         BEGIN                                                       02107
         IF param2 THEN                                              03141
            BEGIN                                                    03142
            CCPOS param2;  param2 ← READC;                           02108
            IF param2 = ENDCHR THEN param2 ← 0;                      03143
            END;                                                     03144
         IF param3 THEN                                              03145
            BEGIN                                                    03146
            CCPOS param3;  param3 ← READC;                           03147
            IF param3 = ENDCHR THEN param3 ← 0;                      03148
            END;                                                     03149
         curmkr ← destination;  curmkr[1] ← 1;                       02110
         CASE param1 OF                                              02111
            = $statement:                                            02112
               csetnsta(destination, param2, param3);                02113
            = $group, = $branch, = $plex:                            02114
               csetngro(destination,  ( DESTINATION*D2sel),
               param2, param3, lda());                               02115
            ENDCASE err(notyet);                                     02116
         END;                                                        02117
      = $private: %file%                                             03778
         chprvsts (lcfile (), $psprivate);                           03779
      = $public: %file%                                              03780
         chprvsts (lcfile (), $pspublic);                            03781
      = $temporary: %modifications to file%                          02153
         csettemmod(lcfile());                                       02154
      = $tenex: %protection for a file%                              04619
         BEGIN                                                       01580
         rhostn ← lnbfls( &param1, 0, $filstr);                      03396
```

```
% parse the input %                                              01584
   CASE [param2 + 1] OF                                         01585
      = $allow:                                                 01586
         BEGIN                                                  01587
         prot ← 0;                                              01588
         IF NOT (count ← [param2] - 2) THEN                     01589
            err($"Illegal protection specified");
                                                                01590
         i ← 1;                                                 01591
         WHILE i <= count DO                                    01592
             prot ← prot .V                                     01593
               (CASE [param2+2+(i := i + 1)] OF
                                                                01594
                  = $read: 40B;                                 01595
                  = $write: 20B;                                01596
                  = $execute: 10B;                              01597
                  = $append: 04B;                               01598
                  = $list: 02B;                                 01599
                  = $all: 77B;                                  01600
                  = $set:                                       01601
                     VALUE([param2+2+(i:=i+1)],8);
                                                                01602
                  ENDCASE 0 );                                  01603
         prot ← prot .A 77B;                                    01604
         CASE [param2 + 2] OF                                   01605
            = $self:                                            01606
               BEGIN                                            01607
               prot ← prot * 10000B;                            01608
               mask ← 770000B;                                  01609
               END;                                             01610
            = $group:                                           01611
               BEGIN                                            01612
               prot ← prot * 100B;                              01613
               mask ← 007700B;                                  01614
               END;                                             01615
            = $public:                                          01616
               BEGIN                                            01617
               prot ← prot * 1B;                                01618
               mask ← 000077B;                                  01619
               END;                                             01620
            ENDCASE                                             01621
               err($"Illegal protection
               specified");                                     01622
         END;                                                   01623
      = $forbid:                                                01624
         BEGIN                                                  01625
         prot ← 0;                                              01626
         IF NOT (count ← [param2] - 2) THEN                     01627
            err($"Illegal protection specified");
                                                                01628
         i ← 1;                                                 01629
         WHILE i <= count DO                                    01630
             prot ← prot .V                                     01631
               (CASE [param2+2+(i := i+1)] OF                   01632
                  = $read: 40B;                                 01633
                  = $write: 20B;                                01634
```

```
                        = $execute: 10B;                    01635
                        = $append: 04B;                     01636
                        = $list: 02B;                       01637
                        = $all: 77B;                        01638
                        = $set:                             01639
                           VALUE(/param2+2+(i:=1+1)/,8);
                                                            01640
                        ENDCASE 0 );                        01641
                    prot ← prot .A 77B .X 77B;              01642
                    CASE /param2 + 2/ OF                    01643
                       = $self:                             01644
                         BEGIN                              01645
                         prot ← prot * 10000B;              01646
                         mask ← 770000B;                    01647
                         END;                               01648
                       = $group:                            01649
                         BEGIN                              01650
                         prot ← prot * 100B;                01651
                         mask ← 007700B;                    01652
                         END;                               01653
                       = $public:                           01654
                         BEGIN                              01655
                         prot ← prot * 1B;                  01656
                         mask ← 000077B;                    01657
                         END;                               01658
                       ENDCASE                              01659
                         err($"Illegal protection
                         specified");                       01660
                    END;                                    01661
                 = $reset:                                  01662
                    BEGIN                                   01663
                    prot ← 777752B;                         01664
                    mask ← 18M;                             01665
                    END;                                    01666
                 = $private:                                01667
                    BEGIN                                   01668
                    mask ← 18M;                             01669
                    prot ← CASE /param2 + 2/ OF             01670
                       = $self: 770000B;                    01671
                       = $group: 777700B;                   01672
                       = $public: 777777B;                  01673
                       ENDCASE 777752B;                     01674
                    END;                                    01675
                 = $set:                                    01676
                    BEGIN                                   01677
                    mask ← 18M;                             01678
                    prot ← VALUE(/param2 + 2/,8);           01679
                    END;                                    01680
                 ENDCASE                                    01681
                    err($"Illegal Protection Specified");
                                                            01682
              *lit* ← NULL;                                 01683
              cprofil(rhostn, $filstr, mask, prot, $lit);   01684
              IF lit.L THEN                                 01685
                 *lit* ← "The protection of the following files
                 has been changed:", EOL, *lit*             01686
```

```
                ELSE                                                    01687
                   *lit*  ← "No files' protection changed";             01688
                fbctl( typecalit, $lit);                                01689
                END;                                                    01690
             = $tty: %window%                                           02155
                BEGIN                                                   02156
                %restore any suppressed da's%                           02157
                   IF (&da ← ttyda) NOT= &msgda AND da.daexis THEN
                                                                        04493
                      BEGIN                                             02161
                      da.daseq ← da.dasuppress ← FALSE;                 02162
                      da.dahandle ← alocda(&da);                        02163
                      END;                                              02164
                &da ← param1;                                           02165
                IF da.dahandle THEN                                     03748
                   BEGIN                                                03765
                   clearda(&da);                                        03762
                   dealocda(&da);                                       03763
                   da.dahandle ← 0;                                     03766
                   END;                                                 03764
                %delete old tty-sim%                                    03749
                   IF ttysim := 0 THEN dealocda(ttyda);                 03750
                %allocate new tty-sim area%                             02173
                   da.daseq ← da.dasuppress ← TRUE;                     02174
                   end1 ← da.dalsz :=
                   (da.dabottom-da.datop)/da.davinc;                    02175
                   save ← linkcnsl := 0;                                02176
                   ttysim ← alocda(ttyda ← &da);                        02177
                   linkcnsl ← save;                                     02178
                   da.dalsz ← end1;                                     02179
                   ttywindow(&da);                                      03751
                dpset(dspyes, da.dacsp, endfil, endfil);                02187
                defttysim ← FALSE;                                      02188
                END;                                                    02189
             = $viewspecs:                                              02190
                BEGIN                                                   02191
                cspupdate ← lda();                                      02192
                cspvs ← param1;                                         02193
                cspvs[1] ← param1[1];                                   02194
                dpset(dspyes, [cspupdate].dacsp, endfil, endfil);
                                                                        02195
                END;                                                    02196
             ENDCASE err(notyet);                                       02197
          END;                                                          02198
       ENDCASE;                                                         02199
    RETURN(&result);                                                    02200
    END.
                                                                        02201
    (setcharsize) % sets all fields in the da for specifed char
    size %                                                              03154
       PROCEDURE ( da, charsize);                                       03156
       LOCAL hinc, vinc;                                                03160
       REF da;                                                          03159
       dpset(dspallf, da.dacsp, endfil, endfil);                        02035
       CASE charsize OF                                                 02040
          = 0:                                                          02041
```

```
                BEGIN                                                   02042
                hinc ← hinc0*256;                                       02043
                vinc ← vinc0*256;                                       02044
                END;                                                    02045
             = 1:                                                       02046
                BEGIN                                                   02047
                hinc ← hinc1*256;                                       02048
                vinc ← vinc1*256;                                       02049
                END;                                                    02050
             = 2:                                                       02051
                BEGIN                                                   02052
                hinc ← hinc2*256;                                       02053
                vinc ← vinc2*256;                                       02054
                END;                                                    02055
             = 3:                                                       02056
                BEGIN                                                   02057
                hinc ← hinc3*256;                                       02058
                vinc ← vinc3*256;                                       02059
                END;                                                    02060
             ENDCASE err($"illegal character size");                    02061
          da.dacsize ← da.danocs ← da.dasgcs ← charsize;                02062
          da.daind ←                                                    02063
             MIN((da.daind/da.daninc)*hinc, da.damind);                 02064
          da.dahinc ← da.danohi ← da.dasghi ← hinc;                     02065
          da.davinc ← vinc;                                             02066
                                                                        02067
          % *** Updatethis from NIC-NLS ***%                           02068
                                                                        02069
          %deallocate old da and get new one%                           02070
             IF nldevice NOT= lineprocessor THEN                        02071
                BEGIN                                                   02072
                r1 ← da.dahandle;                                       02073
                IF NOT SKIP !JSYS dda THEN                              02074
                   err($"DDA JSYS failed, qgdf");                       02075
                da.dahandle ← 0;                                        02076
                IF r1 ← da.dalhandle := 0 THEN                          02077
                   BEGIN                                                02078
                   r2 ← linkcns1;                                       02079
                   IF NOT SKIP !JSYS ndda THEN                          02080
                      dismes(2, $"Link NDDA JSYS failed, qgdf");        02081
                   END;                                                 02082
                END;                                                    02083
             alocda(&da);                                               02084
          RETURN;                                                       03158
          END.                                                          03157
%show%                                                                  02202
   (xshow) %Execute show Command%                                       02203
      PROCEDURE                                                         02204
         %FORMALS%                                                      02205
            (result,          %result record%                           02206
            parsemode,        %parsing, backup, cleanup%                02207
            entity,           %entity type%                             02208
            param,  %param pointer%                                     02209
            dopt,   %directory options for show directory%              02210
            difile);          %filelink for show directory%             02211
         LOCAL                                                          02212
```

```
        rhostn, dskcnt,                                                 03633
        % stuff for show directory %                                    02213
           info,        % record saying what was requested %
                                                                        02214
           gropk,       % record saying how to group things %
                                                                        02215
           sortk        % record saying how to wort things %
                                                                        02216
        ;                                                               02217
     LOCAL STRING filstr[200];                                          03634
     REF                                                                02227
        result, entity, param, dopt, dlfile;                            02228
  %----------------------%                                             02229
  CASE parsemode OF                                                     02230
     = parsing:                                                         02231
        BEGIN                                                           02232
        *lit* ← NULL;  %used for building status messages%              02233
        CASE entity OF                                                  02234
           = $archive: %directory%                                      02235
             cshoarcdir(&param, &param+d2sel, $lit);                    02236
           = $directory:                                                02237
             BEGIN                                                      02238
             info ← gropk ← sortk ← 0;                                  02239
             *filstr* ← "*.*;*";                                        03635
             xdiropt( &param, dopt, &dlfile,                            02240
                $info, $gropk, $sortk, $rhostn, $filstr);               02241
             cshodir(info, gropk, sortk, rhostn, $filstr);              02243
             END;                                                       02245
           = $disk: %space status% %uses connected directory%
                                                                        02246
             IF (dskcnt ← cshodskspa($lit)) > 0 THEN                    02247
                BEGIN                                                   03650
                !gjinf();                                               03652
                gdname( r2, $filstr);                                   03653
                *filstr* ←                                              03654
                   *filstr*, " OVER ALLOCATION BY ",
                   STRING(dskcnt), " PAGES!";                           03656
                dismes(2, $filstr);                                     03657
                END;                                                    03651
           = $file: %status%                                            02248
             CASE param OF                                              02249
                = $status: %all%                                        02250
                  cshofilsta(7, lcfile(), $lit);                        02251
                = $default: %dir for links%                             02252
                  cshofilsta(4, lcfile(), $lit);                        02253
                = $marker: %list%                                       02254
                  cshomarfil(lcfile(), $lit);                           02255
                = $modifications: %status%                              02256
                  cshofilsta(2, lcfile(), $lit);                        02257
                = $return: %ring%                                       02258
                  cshofrring(lda(), $lit);                              02259
                = $size:                                                02260
                  cshofilsta(1, lcfile(), $lit);                        02261
                ENDCASE err(notyet);                                    02262
           = $return: %ring status%                                     04341
             cshosrring( lda(), $lit );                                 04342
```

```
            = $name: %delimiters for statement%                    02263
               cshonsta(param, $lit);                              02264
            = $viewspecs:                                          02265
               xshoviespe(lda(), $lit);                            02266
            ENDCASE err(notyet);                                   02267
         CASE entity OF                                            02268
            = $directory: NULL;  %does its own display%            02269
            ENDCASE fbctl( typecalit, $lit );                      02270
         dpset(dspno, endfil, endfil, endfil);                     02271
         END;                                                      02272
      ENDCASE;                                                     02273
   RETURN(&result);                                                02274
   END.                                                            02275

(xshoviespe)  PROCEDURE (da, string);                              02276
   LOCAL vs[2];                                                    03150
   REF da, string;                                                 02277
   vs ← da.davspec;                                                02278
   vs[1] ← da.davspc2;                                             03151
   curvsp($vs, &string);                                           03152
   RETURN;                                                         02279
   END.                                                            02280

% utility for show directory - prints one file %                   02281
   (xdlpnt)      % print a file for show directory %                02282
      PROCEDURE                                                    02283
         (function,      % requested function %                     02284
         astr            % address of string to be printed %       02285
         );                                                        02286
      REF astr;                                                    02287
                                                                   02288
      CASE function OF                                             02289
         = typenulllit:                                            02290
            fbctl( typenulllit, &astr );                           02291
         = fbaddlit:                                               02292
            fbctl( fbaddlit, &astr );                              02293
         = typecalit:                                              02294
            fbctl( addcalit, &astr );                              02295
         ENDCASE;                                                  02296
      RETURN;                                                      02297
                                                                   02298
      END.                                                         02299

%sort%                                                             02320
   (xsort) %Execute Sort Command%                                  02321
      PROCEDURE                                                    02322
         %FORMALS%                                                 02323
         (result,        %result record%                           02324
         parsemode,      %parsing, backup, cleanup%                02325
         entity,         %entity type%                             02326
         destination); %destination pointer%                       02327
         REF                                                       02328
            result, entity, destination;                           02329
      %----------------------%                                     02330
      CASE parsemode OF                                            02331
         = parsing:                                                02332
```

```
            CASE entity OF                                                  02333
               = $group, = $plex:                                           02334
                  BEGIN                                                     02335
                  csorgro(&destination, &destination+d2sel);                02336
                  curmkr ← gethed(destination); curmkr[1] ← 1;              02337
                  dpset(dspstrc, destination, endfil, endfil);              02338
                  END;                                                      02339
               = $branch:                                                   02340
                  BEGIN                                                     02341
                  IF (destination := getsub(destination)) =
                  destination THEN err($"Illegal Sort");                    02342
                  destination[d2sel] ← getail(destination);                 02343
                  REPEAT CASE($group);                                      02344
                  END;                                                      02345
               ENDCASE err(notyet);                                         02346
            ENDCASE;                                                        02347
         RETURN(&result);                                                   02348
         END.
                                                                            02349
%split%                                                                     02350
%stop%                                                                      02387
   (xstop) %Execute stop Command%                                           02388
      PROCEDURE                                                             02389
         %FORMALS%                                                          02390
            (result,          %result record%                               02391
            parsemode);       %parsing, backup  cleanup%                    02392
            REF                                                             02393
               result;                                                      02394
         %----------------------%                                           02395
         CASE parsemode OF                                                  02396
            = parsing:                                                      02397
               ctlquit();  %ctlquit is also called from HALT%               02398
            ENDCASE;                                                        02399
         RETURN(&result);                                                   02400
         END.
                                                                            02401
%substitute%                                                                02402
   (xsubstitute) %Execute Substitute Command%                               02403
      PROCEDURE                                                             02404
         %FORMALS%                                                          02405
            (result,          %result record%                               02406
            parsemode,        %parsing, backup, cleanup%                    02407
            textentity,       %text entity type%                            02408
            structentity  %structure entity type%                           02409
            destination,      %destination pointer%                         02410
            pairs,  %address of pairs list%                                 02411
            filterflag,       %TRUE if filter requested%                    02412
            vs);              %viewspecs for filter%                        02413
            REF                                                             02414
               result, structentity, destination, textentity, pairs,
               filterflag, vs;                                              02415
         LOCAL savel, save2, savca, savus, da, adstr[40];  REF da;
                                                                            02416
         %----------------------%                                           02417
         CASE parsemode OF                                                  02418
            = parsing:                                                      02419
```

```
        BEGIN                                                          02420
        &da ← lda();                                                   02421
        curmkr ← destination;  curmkr[1] ← 1;                          02422
        savel ← da.davspec;                                            02423
        save2 ← da.davspc2;                                            02424
        savca ← da.dacacode;                                           03857
        savus ← da.dausqcod;                                           03878
        ON SIGNAL ELSE                                                 03859
           BEGIN                                                       03860
           da.davspec ← savel;                                         03862
           da.davspc2 ← save2;                                         03863
           da.dacacode ← savca;                                        03864
           da.dausqcod ← savus;                                        03865
           END;                                                        03861
        IF filterflag THEN                                             02425
           BEGIN                                                       02426
           da.davspec ← vs.vsl;                                        02427
           da.davspc2 ← vs.vs2;                                        02428
           da.dacacode ← vs.vscacode;                                  03868
           da.dausqcod ← vs.vsusqcod;                                  03869
           END                                                         02429
        ELSE                                                           03870
           BEGIN                                                       03871
           da.davspec ← 0;                                             03878
           da.davspec.vslev ← da.davspec.vstrnc ←                      03872
              da.davspec.vsnamf ← da.davspec.vsdaft ←                  03879
                 da.davspec.vsindf ← -1;                               03880
           da.dacacode ← da.dausqcod ← 0;                              03874
           END;                                                        03876
        CASE structentity OF                                           02430
           = $statement:                                               02431
              BEGIN                                                    02432
              dpset(dsprfmt, destination, endfil, destination);        02433
              clist(ctcmk, destination.stfile, nofile);                02434
              csubsta(destination, pairs, &da);                        02435
              clupdt();                                                02436
              END;                                                     02437
           = $group, = $plex, = $branch:                               02438
              BEGIN                                                    02439
              dpset(dspallf,destination,endfil,endfil);                02440
              clist(ctcmk, destination.stfile, nofile);                02441
              csubgro(destination, [&destination+d2sel], pairs,        02442
              &da);                                                    02443
              clupdt();                                                02444
              END;                                                     02445
           ENDCASE err(notyet);                                        02446
        da.davspec ← savel;                                            02447
        da.davspc2 ← save2;                                            03866
        da.dacacode ← savca;                                           03867
        da.dausqcod ← savus;                                           02448
        END;                                                           02449
     ENDCASE;                                                          02450
  RETURN(&result);                                                     02451
  END.
```

```
%transpose%                                                          02452
   (xtranspose) %Execute Transpose Command%                          02453
      PROCEDURE                                                      02454
         %FORMALS%                                                   02455
            (result,          %result record%                       02456
            parsemode,        %parsing, backup, cleanup%            02457
            sourcentity,      %source entity type%                  02458
            source,           %source pointer%                      02459
            destentity,       %destination entity type%             02460
            destination,      %destination pointer%                 02461
            filterflag,       %if TRUE, filtered with viewspecs in vs%
                                                                     02462
            vs);              %viewspec string%                     02463
            REF                                                      02464
               result, sourcentity, source, destentity, destination,
               filterflag, vs;                                       02465
         LOCAL adstr[40];  % block for parsing links %               03322
      %-----------------------%                                      02466
      CASE parsemode OF                                              02467
         = parsing:                                                  02468
            BEGIN                                                    03300
            CASE sourcentity OF                                      03302
               = $link:                                              03303
                  BEGIN                                              03305
                  IF source.stastr THEN                              03454
                     BEGIN                                           03455
                     IF NOT FIND                                     03456
                        SF(source) $(SP/TAB) ('(/'</"--") THEN       03457
                           ST source ← '<, SF(source) SE(source);
                                                                     03458
                     IF NOT FIND                                     03459
                        SE(source) $(SP/TAB) (')/'>) THEN            03460
                           ST source ← SF(source) SE(source), '>;
                                                                     03461
                     END;                                            03462
                  lnkprs( &source, $adstr);                          03306
                  source ← adstr[ls];                                03307
                     source[1] ← adstr[ls+1];                        03318
                  [&source+d2sel] ← adstr[le];                       03308
                     [&source+d2sel+1] ← adstr[le+1];                03319
                  END;                                               03309
               ENDCASE;                                              03304
            CASE  DESTENTITY OF                                       03310
               = $link:                                              03311
                  BEGIN                                              03312
                  IF destination.stastr THEN                         03463
                     BEGIN                                           03464
                     IF NOT FIND                                     03465
                        SF(destination) $(SP/TAB) ('(/'</"--") THEN
                                                                     03466
                           ST destination ←                          03467
                              '<, SF(destination) SE(destination);
                                                                     03472
                     IF NOT FIND                                     03468
                        SE(destination) $(SP/TAB) (')/'>) THEN       03469
                           ST destination ←                          03470
```

```
                        SF(destination) SE(destination), '>;
                                                                    03473
                    END;                                            03471
                  lnkprs( &destination, $adstr);                    03313
                  destination ← adstr[ls];                          03314
                    destination[1] ← adstr[ls+1];                   03320
                  [&destination+d2sel] ← adstr[le];                 03315
                    [&destination+d2sel+1] ← adstr[le+1];           03321
                  END;                                              03316
                ENDCASE;                                            03317
              CASE sourcentity OF                                   02469
                = $character, = $invisible, = $text, = $word, =
                $visible, = $number, = $link:                       02470
                  BEGIN                                             02471
                  clist (ctcmk, destination.stfile, source.stfile);
                                                                    02472
                  dpset(dsprfmt, destination, source, endfil);      02473
                  curmkr ← source;  curmkr[1] ← source[1];          02474
                  ctratex(&destination, &destination+d2sel, &source,
                  &source+d2sel);                                   02475
                  clupdt ();                                        02476
                  END;                                              02477
                = $statement:                                       02478
                  BEGIN                                             02479
                  clist (ctcsp, destination.stfile, source.stfile);
                                                                    02480
                  dpset(dspstrc, destination, source, endfil);      02481
                  curmkr ← destination;  curmkr[1] ← 1;             02482
                  ctrasta(destination, source, filterflag, &vs);
                                                                    02483
                  clupdt ();                                        02484
                  END;                                              02485
                = $group, = $plex, = $branch:                       02486
                  BEGIN                                             02487
                  clist (ctcsp, destination.stfile, source.stfile);
                                                                    02488
                  dpset(dspstrc,destination,source,endfil);         02489
                  curmkr ← source;  curmkr[1] ← 1;                  02490
                  ctragro(destination, destination[d2sel], source,
                  source[d2sel], filterflag, &vs);                  02491
                  clupdt ();                                        02492
                  END;                                              02493
                ENDCASE err(notyet);                                02494
            END;                                                    03301
          ENDCASE;                                                  02495
        RETURN(&result);                                            02496
        END.
                                                                    02497
%trim%                                                              02498
    (xtrim) %Execute Trim Command%                                  02499
        PROCEDURE                                                   02500
            % FORMALS%                                               02501
            (result,        %result record%                         02502
            parsemode,      %parsing, backup, cleanup%              02503
            parameter);     %number of versions%                    02504
            REF                                                     02505
```

```
            result, parameter;                                          02506
        LOCAL tlngth; % temporary for string length %                   02507
        %-----------------------%                                       02508
        CASE parsemode OF                                               02509
           = parsing:                                                   02510
              BEGIN                                                     02511
              result ← 0;                                               02512
              result ← getstring(3000, $dspblk);                        02513
              */result/* ← "Trimmed Files Are:
              ";                                                        02514
              tlngth ← /result/.L;                                      02515
              ctridir(getpint(&parameter, &parameter+d2sel), result);
                                                                        02516
                 %trim connected directory%                             02517
              IF ( /result/.L > tlngth ) THEN                           02518
                 fbctl( typecalit, result)                              02519
              ELSE fbctl( typecalit, $"No Files Trimmed" );             02520
              END;                                                      02521
           = backup, = cleanup:                                         02522
              IF result THEN freestring(result, $dspblk);               02523
           ENDCASE;                                                     02524
        RETURN(&result);                                                02525
        END.
                                                                        02526
%undelete%                                                              02527
   (xundelete) %Execute Undelete Command%                               02528
        PROCEDURE                                                       02529
           %FORMALS%                                                    02530
              (result,          %result record%                         02531
              parsemode,        %parsing, backup, cleanup%              02532
              entity,           %entity type%                           02533
              filename);        %filename pointer%                      02534
              REF                                                       02535
                 result, entity, filename;                              02536
           LOCAL stid, tlngth, rhostn;                                  02537
           LOCAL STRING filestr/200/;                                   03294
        %-----------------------%                                       02540
        CASE parsemode OF                                               02541
           = parsing:                                                   02542
              BEGIN                                                     02543
              result ← 0;                                               02544
              CASE entity OF                                            02545
                 = $file:                                               02546
                    BEGIN                                               02547
                    result ← getstring(3000, $dspblk);                  02548
                    */result/* ← "Undeleted Files Are:
                    ";                                                  02549
                    tlngth ← /result/.L;                                02550
                    rhostn ← lnbfls( &filename, 0, $filestr);           03295
                    cundfil(rhostn, $filestr, result);                  02554
                    IF ( /result/.L > tlngth ) THEN                     02555
                       fbctl( typecalit, result)                        02556
                    ELSE fbctl( typecalit, $"No Files Undeleted" );
                                                                        02557
                    END;                                                02558
                 = $archive: %file%                                     02559
```

```
                cundarcfil(&filename, &filename+d2sel);
              = $modifications: %to file%
                BEGIN
                stid ← orgstid;
                stid.stfile ← lcfile();
                clist (ctlcfm, stid.stfile, nofile);
                dpset(dspallf, stid, endfil, endfil);
                cundmodfil(stid.stfile);
                <IOEXEC, unlkclist> (); %check clist items%
                clupdt ();
                END;
              ENDCASE err(notyet);
           END;
        = backup, = cleanup:
           IF result THEN freestring(result, $dspblk);
        ENDCASE;
     RETURN(&result);
     END.

%update%
  (xupdate) %Execute Update Command%
     PROCEDURE
        %FORMALS%
           (result,          %result record%
           parsemode,        %parsing, backup, cleanup%
           entity,           %entity type%
           filename);        %filename pointer%
        LOCAL tp2, fileno, stid;
        LOCAL STRING filstring[200];
        REF
           result, entity, filename, tp2;
        %----------------------%
        CASE parsemode OF
           = parsing:
              BEGIN
              stid ← orgstid;
              stid.stfile ←  fileno ← lcfile();
              dpset(dsprfmt, stid, endfil, endfil);
              CASE entity OF
                 = $old:
                    cupdfil (fileno, oldversion, 0);
                 = $new:
                    cupdfil (fileno, newversion, 0);
                 = $compact: %file%
                    BEGIN
                    dpset(alldspf, stid, endfil, endfil);
                    clist(15, fileno, 0);
                    cupdfil (fileno, upcompact, 0);
                    clupdt();
                    END;
                 = $rename: %file%
                    BEGIN
                    % move file name to local string %
                       CASE lnbfls( &filename, 0, $filstring) OF
                          = lhostn: NULL;
                          ENDCASE
```

```
                        err($"Remote File Manipulations Not
                        Implemented Yet");                              03293
                    cupdfil (fileno, newname, $filstring);              02616
                    END;                                                02617
                ENDCASE err(notyet);                                    02618
            *filstring* ← NULL;                                         02623
            filnam( fileno, $filstring);                                02624
            dismes( 2, $filstring);                                     02625
            END;                                                        02626
        ENDCASE;                                                        02627
    RETURN(&result);                                                    02628
    END.
                                                                        02629
%verify%                                                                02630
  (xverify) %Execute Verify Command%                                    02631
    PROCEDURE                                                           02632
      %FORMALS%                                                         02633
        (result,          %result record%                               02634
        parsemode);       %parsing, backup, cleanup%                    02635
        REF                                                             02636
          result;                                                       02637
    %----------------------%                                            02638
    CASE parsemode OF                                                   02639
      = parsing:                                                        02640
        BEGIN                                                           04482
        cverfil(lcfile(), TRUE);                                        02641
        dismes(1, $"Successful: internal structure is OK");             04484
        END;                                                            04483
      ENDCASE;                                                          02642
    RETURN(&result);                                                    02643
    END.
                                                                        02644
%TAB%                                                                   02645
  (xtab) PROCEDURE( % Execute repeat last search command %              02646
    % FORMAL ARGUMENTS %                                               02647
      resultptr,          % ptr to result record %                     02648
      parsemode);         % interpreter parsing mode %                 02649
      REF resultptr=:
    local da;  REF da;                                                  02651
    LOCAL TEXT POINTER t1, t2;                                          02652
    LOCAL STRING treps[200];                                            02653
    CASE parsemode OF                                                   02654
      = parsing:                                                        02655
        BEGIN                                                           02656
        CASE srchtype OF                                                02657
          = wordtyp:                                                    02658
            *treps* ← "=w";                                             03167
          = wordls:                                                     03170
            *treps* ← "=ws";                                            03171
          = contnt:                                                     02663
            *treps* ← "=c";                                             03168
          = contls:                                                     02668
            *treps* ← "=cs";                                            03169
          ENDCASE err($"<tab> valid only to repeat a previous
          search");                                                     02670
        *treps* ← ".n", ', *conreg*, ', *treps*;                        02671
```

```
            dismes(1, $treps);                                          02672
            FIND SF(*treps*) ↑t1 SE(*treps*) ↑t2;                       02674
            caddexp($t1, $t2, cspupdate ← lda(), $curmkr);              02675
            da.dacsp ← curmkr; da.daccnt ← curmkr[1];                   02676
            dpset(dspjpf, curmkr, endfil, endfil);                      02677
            dismes(0,0);                                                03185
            END;                                                        02678
         ENDCASE;                                                       02679
      RETURN (&resultptr);                                              02680
      END.
                                                                        02681
%<LF>%                                                                  02706
   (xlinefeed) PROCEDURE( % Execute TNLS print next statement %         02707
      % FORMAL ARGUMENTS %                                             02708
         resultptr,          % ptr to result record %                  02709
         parsemode);         % interpreter parsing mode %              02710
         REF resultptr;                                                 02711
      LOCAL stid, da;  REF da;                                          02712
      LOCAL lvs[2];          % viewspec param record %                  02713
      LOCAL TEXT POINTER t1, t2;                                        02714
      LOCAL STRING string[5];                                           02715
      %----------%                                                      02716
      CASE parsemode OF                                                 02717
         = parsing:                                                     02718
            BEGIN                                                       02719
            *string* ←".n";                                             02720
            FIND SF(*string*) ↑t1 SE(*string*) ↑t2;                     02721
            caddexp($t1, $t2, &da ← lda(), $curmkr);                    02722
            lvs ← da.davspec;                                           02723
            lvs[1] ← da.davspc2;                                        02724
            cprista( curmkr, $lvs, &da );                               02725
            dpset(dspjpf, curmkr, endfil, endfil);                      02726
            END;                                                        02727
         ENDCASE;                                                       02728
      RETURN (&resultptr);                                              02729
      END.
                                                                        02730
%↑%                                                                     02731
   (xuparrow) PROCEDURE( % Execute TNLS print previous statement %
                                                                        02732
      % FORMAL ARGUMENTS %                                             02733
         resultptr,          % ptr to result record %                  02734
         parsemode);         % interpreter parsing mode %              02735
         REF resultptr;                                                 02736
      LOCAL stid, da;  REF da;                                          02737
      LOCAL lvs[2];          % viewspec param record %                  02738
      LOCAL TEXT POINTER t1, t2;                                        02739
      LOCAL STRING string[5];                                           02740
      %----------%                                                      02741
      CASE parsemode OF                                                 02742
         = parsing:                                                     02743
            BEGIN                                                       02744
            *string* ←".b";                                             02745
            FIND SF(*string*) ↑t1 SE(*string*) ↑t2;                     02746
            caddexp($t1, $t2, &da ← lda(), $curmkr);                    02747
            lvs ← da.davspec;                                           02748
```

```
            lvs[1] ← da.davspc2;                                        02749
            cprista( curmkr, $lvs, &da );                               02750
            dpset(dspjpf, curmkr, endfil, endfil);                      02751
            END;                                                        02752
         ENDCASE;                                                       02753
      RETURN (&resultptr);                                              02754
      END.                                                              02755

%.%                                                                     02756
   (xperiod) PROCEDURE( % Execute TNLS print current location %         02757
      % FORMAL ARGUMENTS %                                             02758
         resultptr,          % ptr to result record %                   02759
         parsemode);         % interpreter parsing mode %               02760
         REF resultptr;                                                 02761
      LOCAL STRING astrng[50]; % collection string %                    02762
      %----------%                                                      02763
      CASE parsemode OF                                                 02764
         = parsing:                                                     02765
            BEGIN                                                       02766
            cspupdate ← FALSE;                                          02767
            ccurloc($curmkr, $astrng );                                 02768
            typeas( $astrng );                                          02769
            END;                                                        02770
         ENDCASE;                                                       02771
      RETURN (&resultptr);                                              02772
      END.                                                              02773

%/%                                                                     02774
   (xslash) PROCEDURE( % Execute TNLS / command %                       02775
      % FORMAL ARGUMENTS %                                             02776
         resultptr,          % ptr to result record %                   02777
         parsemode);         % interpreter parsing mode %               02778
         REF resultptr;                                                 02779
      LOCAL STRING string[100];                                         02780
      CASE parsemode OF                                                 02781
         = parsing:                                                     02782
            BEGIN                                                       02783
            cspupdate ← FALSE;                                          02784
            ccurcon( $curmkr, $string );                                02785
            typeas($string);                                            02786
            END;                                                        02787
         ENDCASE;                                                       02788
      RETURN (&resultptr);                                              02789
      END.                                                              02790

%\%                                                                     02791
   (xbslash) PROCEDURE( % Execute TNLS \ command %                      02792
      % FORMAL ARGUMENTS %                                             02793
         resultptr,          % ptr to result record %                   02794
         parsemode);         % interpreter parsing mode %               02795
         REF resultptr;                                                 02796
      LOCAL da, csp, cnt;                                               02797
      LOCAL lvs[2];          % viewspec param record %                  02798
      REF da;                                                           02799
      CASE parsemode OF                                                 02800
         = parsing:                                                     02801
```

```
            BEGIN                                                   02802
            cspupdate ← FALSE;                                      02803
            csp ← curmkr;  cnt ← curmkr[1];                         02804
            &da ← lda();                                            02805
            lvs ← da.davspec;                                       02806
            lvs[1] ← da.davspc2;                                    02807
            cprista( curmkr, $lvs, &da );                           02808
            curmkr ← csp;  curmkr[1] ← cnt;                         02809
            END;                                                    02810
         ENDCASE;                                                   02811
      RETURN (&resultptr);                                          02812
      END.
                                                                    02813
% Substitute support routines %                                    02941
   (subsinit) PROC( % initializes state for the substitute command %
                                                                    02942
      % FORMAL ARGUMENTS %                                         02943
         resultptr,     % ptr to result record %                   02944
         parsemode,     % parsing mode %                           02945
         textent);      % text entity type for the substitute %    02946
      LOCAL % VARIABLES %                                           02947
         subdsp,        % ptr to substitute display buffer %       02948
         ttype;         % text entity code %                       02949
      LOCAL STRING subm[30];  % string for "Subs = " message %      02950
      REF subdsp, resultptr, textent;                               02951
      % ---------- %                                               02952
      CASE parsemode OF                                             02953
         = parsing:                                                 02954
            BEGIN                                                   02955
            % set ttype according to type of text entity-- This code
            should be removed after sbinit is changed to expect the new
            type of text entity codes %                             02956
               ttype ← CASE textent OF                              02957
                  = $character:         charv;                      02958
                  = $word:              wordv;                      02959
                  = $number:            numbrv;                     02960
                  = $visible:           visv;                       02961
                  = $invisible:         invisv;                     02962
                  = $link:              linkv;                      02963
                  = $text:              textv;                      02964
                  ENDCASE err( notyet );                            02965
            % allocate a display buffer for the substitute processor %
                                                                    02966
               resultptr ← resultptr[1] ← 0; %will be used in cleanup.
               %                                                    03702
               resultptr ← &subdsp ← getstring( 640, $dspblk );     02967
            % allocate an astr buffer for the substitute processor %
                                                                    03699
               resultptr[1] ← getstring( 500, $dspblk );            03698
            % initialize the substitute state info %               02968
               sbinit( $subhed, $subrec, &subdsp, resultptr[1], ttype
               );                                                   02969
            % initialize substitutions count %                     02977
               subcnt ← 0;                                          02978
            END;                                                    02979
         = backup,                                                  02980
```

```
      = cleanup:                                                    02981
         BEGIN                                                      02982
         % deallocate the display buffer if allocated %             02983
            IF resultptr THEN freestring( resultptr, $dspblk );     02984
         % deallocate the astr if allocated %                       03700
            IF resultptr[1] THEN freestring( resultptr[1], $dspblk
            );                                                      03701
         % display the nuber of substitutions made %                02985
            *subm* ← "Substitutions made: ", STRING(subcnt);        02986
            dismes(2, $subm);                                       02987
         END;                                                       02993
      ENDCASE;                                                      02994
   RETURN (&resultptr ); % TRUE return %                            02995
   END.
                                                                    02996
(sbinit) PROCEDURE(sbhed, sbrec, sbdsp, sbastr, sbttype);           02997
   % initialize substitute data structure %                        02998
   LOCAL j;                                                         02999
   POINTER sbhed;                                                   03000
   sbhed.sbrp ← sbrec;                                              03001
   sbhed.sbdp ← sbdsp;                                              03002
   sbhed.sbas ← sbastr;                                             03003
   sbhed.sbtt ← sbttype;                                            03004
   *[sbastr]* ← NULL;                                               03005
   FOR j ← 0 UP UNTIL >=200B DO [sbdsp+j] ← 0;                      03006
   RETURN;                                                          03007
   END.
                                                                    03008
                                                                    03009
(subldsp) PROCEDURE( % substitute prompting function %              03010
   % FORMAL ARGUMENTS %                                             03011
      resultptr,     % ptr to the result record %                   03012
      parsemode,     % parsing mode %                               03013
      textent,       % text entity type for the substitute %        03014
      newflag);      % TRUE if new text entity is being collected %
                                                                    03015
   LOCAL STRING string[25];          % prompting string %           03016
   REF resultptr, textent, newflag;                                 03017
   % ---------- %                                                   03018
   CASE parsemode OF                                                03019
      = parsing:                                                    03020
         BEGIN                                                      03021
         IF nlmode = typewriter THEN crlf();                        03022
         IF newflag                                                 03023
            THEN *string* ← "New "                                  03024
            ELSE *string* ← "Old ";                                 03025
         *string* ← *string*, *[textent]*;                          03026
         fbctl( echostr, $string );                                 03027
         END;                                                       03028
      ENDCASE;                                                      03029
   RETURN (&resultptr);                                             03030
   END.
                                                                    03031
(sub2dsp) PROCEDURE( % substitute prompting function %              03032
   % FORMAL ARGUMENTS %                                             03033
      resultptr,     % ptr to the result record %                   03034
```

```
    parsemode,     % parsing mode %                                    03035
    type,                  % type of selection made %                  03324
    tptr);         % ptr to string tptr record %                       03036
  LOCAL tptr2, adstr[40];                                              03037
  LOCAL STRING string[100];          % prompting string %              03038
  REF resultptr, tptr, tptr2, type;                                    03039
  % ------------ %                                                     03040
  CASE parsemode OF                                                    03041
    = parsing:                                                         03042
      BEGIN                                                            03043
      CASE type OF                                                     03325
        = $link:                                                       03326
          BEGIN                                                        03327
          IF (tptr.stastr) AND                                         03337
            ( NOT FIND SF(tptr) $(SP/TAB) ('(/'</"--") ) THEN
                                                                       03338
              ST tptr ← '(, SF(tptr) SE(tptr);                         03339
          IF (tptr.stastr) AND                                         03340
            ( NOT FIND SE(tptr) $(SP/TAB) (')/'>) ) THEN               03341
              ST tptr ← SF(tptr) SE(tptr), '>;                         03342
          lnkprs( &tptr, $adstr);                                      03328
          tptr ← adstr[ls];                                            03329
            tptr[1] ← adstr[ls+1];                                     03330
          [&tptr+d2sel] ← adstr[le];                                   03331
            [&tptr+d2sel+1] ← adstr[le+1];                             03332
          END;                                                         03333
        ENDCASE;                                                       03334
      IF nlmode = fulldisplay                                          03044
        AND NOT tptr.stastr THEN                                       03045
          BEGIN                                                        03046
          &tptr2 ← &tptr + d2sel;                                      03047
          *string* ← tptr tptr2;                                       03048
          aplit( $string ); % feedback bugged param %                  03049
          END;                                                         03050
      END;                                                             03051
    ENDCASE;                                                           03052
  RETURN (&resultptr);                                                 03053
  END.
                                                                       03054
(substatus) PROCEDURE (                                                04540

  % The status of a substitute command is printed out before
  execution, given the entity type being substituted%                  04541
  % FORMAL ARGUMENTS %                                                 04542
    resultptr,     % ptr to the result record %                       04543
    parsemode,     % parsing mode %                                   04544
    type);                 % type of selection made %                  04545
  LOCAL pairstr, cardp, wbp1, wbp2, i, j;                              04546
  LOCAL STRING statstr[400], sentstr[10], tempstr[20], oldstr[20],
  newstr[20];                                                          04547
  POINTER cardp;                                                       04548
  REF pairstr, type, resultptr;                                        04549

                                                                       04550
  CASE parsemode OF                                                    04551
    = parsing:                                                         04552
```

```
BEGIN                                                                   04553
%Set up the table heading for output%                                   04554
*sentstr* ← *[type]*;                                                   04555
FOR j ← sentstr.L UP UNTIL = sentstr.M DO                               04556
   *sentstr* ← *sentstr*,SP;                                            04557
*statstr* ← "   SUBSTITUTE ", *sentstr*, EOL, "NEW ",
*sentstr*, "        OLD ", *sentstr*, EOL, EOL;                         04558

%get ahold of the packed string of pairs%                               04559
&pairstr ← subhed.sbas;                                                 04560
                                                                        04561
i ← 1;              %Index through the string%                          04562
WHILE i < pairstr.L DO                                                  04563
   BEGIN                                                                04564
   oldstr.L ← newstr.L ← cardp ← 0;                                     04565
   cardp ← subhed.sbdp + *pairstr*[i];                                  04566
      %get pointer to chain describing entities staring
      with the first character of this entity%                          04567
   IF [cardp] = 0 THEN err($"substatus:  Substitution array
   bad.")                                                               04568
   ELSE cardp ← [cardp];                                                04569
   DO                                                                   04570
      BEGIN                                                             04571
      wbp1 ← chbmty + $oldstr;                                          04572
      wbp2 ← cardp.catbp;                                               04573
      FOR j ← 0 UP UNTIL = cardp.catnc DO                               04574
         BEGIN                                                          04575
         IF j = oldstr.M THEN EXIT;                                     04576
         ↑wbp1 ← ↑wbp2;    %store the old string%                       04577
         END;                                                           04578
      oldstr.L ← MIN(cardp.catnc, j);                                   04579
      *tempstr* ← *pairstr*[i TO oldstr.L+i-1];                         04580
      (comstrs);                                                        04581
      IF *tempstr* = *oldstr* THEN                                      04582
         BEGIN     %This card describes the next pair%                  04583
         wbp1 ← chbmty + $newstr;                                       04584
         wbp2 ← cardp.carbp;                                            04585
         FOR j ← 0 UP UNTIL = cardp.carnc DO                            04586
            BEGIN                                                       04587
            IF j = newstr.M THEN EXIT;                                  04588
            ↑wbp1 ← ↑wbp2;          %store the new string%
                                                                        04589
            END;                                                        04590
         newstr.L ← MIN(cardp.carnc, j);                                04591
         EXIT;                                                          04592
         END                                                            04593
      ELSE cardp ← cardp.canxt;     %next entry%                        04594
      END                                                               04595
   WHILE cardp;                                                         04596
                                                                        04597
   IF oldstr.L = 0 THEN err($"substatus:  String not found
   in array.");                                                         04598
                                                                        04599
   %Add this entry to output status string%                             04600
   FOR j ← newstr.L UP UNTIL = newstr.M DO                              04601
      *newstr* ← *newstr*, SP;                                          04602
```

```
            *statstr* ← *statstr*, *newstr*, SP, SP, *oldstr*, EOL;
                                                                    04603
                                                                    04604
            %Increment i for next pair of strings%                  04605
            i ← i + cardp.catnc + cardp.carnc;                      04606
            END;                                                    04607

         %Send it out and set display globals%                      04608
         fbctl(typecalit, $statstr);                                04609
         dpset(dspno, endfil, endfil, endfil);                      04610

         END;                                                       04611

      ENDCASE;                                                      04612

      RETURN( &resultptr );                                         04613
                                                                    04614
   END.                                                             04615
(subpsave) PROCEDURE( % stashes away parameters for substitute %
                                                                    03055
   % FORMAL ARGUMENTS %                                           03056
      resultptr,     % ptr to the result record %                  03057
      parsemode,     % parsing mode %                              03058
      newptr,        % ptr to the new entity %                     03059
      oldptr);       % ptr to the old entity %                     03060
   LOCAL new2, old2;                                                03061
   LOCAL STRING newstr[200], oldstr[200];                           03062
   REF resultptr, newptr, oldptr, new2, old2;                       03063
   % ------------ %                                                03064
   CASE parsemode OF                                                03065
      = parsing:                                                    03066
         BEGIN                                                      03067
         % fetch parameters to local strings %                     03068
            &new2 ← &newptr + d2sel;                                03069
            &old2 ← &oldptr + d2sel;                                03070
            *newstr* ← newptr new2;                                 03071
            *oldstr* ← oldptr old2;                                 03072
         % check length of the old (target) string %              03073
            IF oldstr.L = empty THEN                                03074
               err( $"cannot substitute for a null text field" );
                                                                    03075
         % stash away the parameter strings %                     03076
            sbpush( $newstr, $oldstr, $subhed );                    03077
         % set return value to be pointer to saved pairs %        03078
            resultptr ← $subhed;                                    03079
         END;                                                       03080
      ENDCASE;                                                      03081
   RETURN (&resultptr);                                             03082
   END.
                                                                    03083
(sbpush) PROCEDURE(str1, str2, hed);                                03084
   %.....Documentation.....%                                        03085
      %The test strings are sorted by initial character and chained
      together, with the head of the chain in a character-code
      indexed array subdsp.  This allows a very fast check whether a
      character can possibly begin a test string.  Each
```

```
      test-replacement pair is represented by a record (card) which
      is linked to others for the same initial test character
      through the canxt field.%                                        03086
      %Global variables used:                                          03087
         subcnt  count of substitutions                                03088
         lit     A-string for building up new statement                03089
         swork   internal work area, see (stbpget)                     03090
      %                                                                03091
   % add pair to substitute list %                                     03092
   LOCAL astr, len, len1, len2, subrp, asa, cap;                       03093
   POINTER hed, subrp, cap;                                            03094
   REF str1, str2, astr;                                               03095
   &astr ← hed.sbas;                                                   03096
   len ← astr.L;                                                       03097
   asa ← &astr + 1; %origin of text for A-string%                      03098
   len1 ← str1.L;                                                      03099
   len2 ← str2.L;                                                      03100
   *astr* ← *astr*, *str2*;                                            03101
   IF hed.sbtt = numbrv THEN                                           03102
      % pad replacement if shorter %                                   03103
      FOR len1 UP UNTIL >=len2 DO                                      03104
         *astr* ← *astr*, SP;                                          03105
   *astr* ← *astr*, *str1*;                                            03106
   subrp ← hed.sbrp;                                                   03107
   subrp.carnc ← len1;                                                 03108
   subrp.catnc ← len2;                                                 03109
   subrp.carbp ← conbp(asa,len+len2);                                  03110
   subrp.catbp ← conbp(asa,len);                                       03111
   subrp.canxt ← 0;                                                    03112
   %link card into appropriate chain%                                  03113
      cap ← hed.sbdp + *str2*[1];                                      03114
      IF [cap] = 0 THEN %empty chain%                                  03115
         [cap] ← subrp                                                 03116
      ELSE BEGIN %link on end%                                         03117
         cap ← [cap];                                                  03118
         WHILE cap.canxt DO cap ← cap.canxt;                           03119
         cap.canxt ← subrp;                                            03120
      END;                                                             03121
   hed.sbrp ← subrp + lcard;                                           03122
   RETURN END.                                                         03123
                                                                       03124
DECLARE bpary = (0106777777777B, 3507B8, 2607B8, 1707B8, 1007B8);
                                                                       03125
(conbp) PROCEDURE(base,cn);                                            03126
   % construct byte pointer assuming 5 characters per word,  cn is
   the character number %                                              03127
   LOCAL q, r;                                                         03128
   DIV cn / 5, q, r;                                                   03129
   RETURN(base+bpary[r]+q);                                            03130
   END.                                                                03131
                                                                       03132
FINISH of psedit                                                       03133
```

PSCALC

PSCALC PSCALC PSCALC PSCALC PSCALC PSCALC PSCALC PSCALC
PSCALC PSCALC PSCALC PSCALC PSCALC PSCALC PSCALC PSCALC

```
< NLS, PSCALC.NLS;11, >, 24-SEP-74 11:11 EKM ;;;;( MICHAEL,
PSCALC.NLS;1, ), 22-MAR-74 02:51 KEV :
FILE pscalc % L10 <REL-NLS>PSCALC.REL %% (110,) (REL-NLS,PSCALC.REL,) %
                                                                        02
% DECLARATIONS %                                                       03
   REF cda;                                                             04
   DECLARE EXTERNAL STRING     %CALCULATOR ERROR MESSAGES%              05
      spliterr = "Need a larger window",                                06
      formdigerr = "too many digits-default format set",                07
      badaccno = "Use a value between 1 and 10",                        08
      caupderr = "System error: Unable to re-open
       calc-ident file",                                                09
      calsyserr = "CALCULATOR SYSTEM ERROR",                           010
      acsaverr = "No saved accumulators found",                        011
      badcfile = "Bad Calc-Ident file, unable to go on",               012
      insizerr = "Format too small for input",                         013
      acsizerr = "Format too small for accumulator
      FORMAT RESET TO DEFAULT",                                        014
      acsizdef = "Format too small for accumulator
      ACCUMULATOR SET TO ZERO",                                        015
      blanks = "                    ";
                                                                       016
   REGISTER                                                            017
      r0 = 0, r1 = 1, r2 = 2, r3 = 3, r4 = 4, r5 = 5, r6 = 6, a1 = 12,
                                                                       018
      a2 = 13, a3 = 14, a4 = 15;
                                                                       019
   (mask) RECORD                                                       022
    0 fld3[6],  # chars in exponent - zero                             023
    2 fld2[6],  # chars after dec - rounds                             024
   10 fld1[6],  # chars front of decimal                               025
   12 round[5],                                                        026
    0 blnk[1],  not used                                               027
    0 oflo[1]    deal with overflow ---> error return                  028
    0 exsign[2], exponent format -00                                   029
    0 exp2[2],  exponent field -00                                     030
    1 dpt[1],   decimal point on/off                                   031
    0 dol[1],   $ on/off                                               032
    1 dig[1],   at least one digit (maybe 0) to left of decimal        033
    1 just[2],  fill field                                             034
    0 sign[2];  0: leave space for + sign
                1: put in + sign                                       035
% CALCULATOR SUBSYSTEM %                                              040
   % INITIALIZATION %
                                                                       041
      (xcalcinit) PROCEDURE (resultptr, parse);                        042
         REF resultptr;                                                043
         LOCAL                                                         044
            fileno;   %file number of calc-ident%                      045
         LOCAL TEXT POINTER tp1, tp2;                                  046
         LOCAL STRING flnam[50];                                       047
         CASE parse OF                                                 048
            = parsing:                                                 049
               BEGIN                                                   050
                  IF cpadent THEN abortsubsystem($"You are already in
                  the calculator")                                     051
```

```
            ELSE cbadent ← TRUE;                                    052
          % load and set up calculator file %                       053
            cafiliinitialize();                                     054
          % clear all accumulators %                                055
            caclear(0, 10);                                         056
          % initialize %                                            057
            asub ← 0;                                               058
            *opstring* ← NULL;                                      059
            *signstr* ← '+;                                         060
            *astrng* ← '1; %default is first accumulator%           061
            %mark beginning of this calculator entry in file%       062
               tp1 ← tp2 ← $"****************";                    063
               tp1.stastr ← 1;                                      064
               tp2.stastr ← 1;                                      065
               tp1[1] ← 1;                                          066
               tp2[1] ← 18;                                         067
               castid ← cinssta(castid,sucdir,$tp1,$tp2);           068
            proc ← $qcadd;                                          069
            qfloutp($accum, $acstring,2);%convert starting accum%
                                                                    070
            CASE nlmode OF                                          071
               = typewriter: typeas($"  0.00");                     072
            ENDCASE                                                 073
               BEGIN                                                074
               dismes(1,$acstring); %display starting accums%       075
               dpset(dspno, endfil, endfil, endfil);                076
               END;                                                 077
         END;                                                       078
       ENDCASE NULL;                                                079
     % exit back to parser/control %                                080
       RETURN(&resultptr);                                          081
     END.
                                                                    082
% FEEDBACK, COMMAND INITIALIZATION %                                083
  (cfeedback) PROCEDURE(result,parsemode); % do calculator feedback
  %                                                                 084
     LOCAL STRING mess[35];
                                                                    085
     REF result;                                                    086
     CASE parsemode OF                                              087
        = parsing:                                                  088
           BEGIN                                                    089
           % floating result to string %                           090
             qfloutp($accum+asub, $acstring,2);                     091
           % see if special formatting char, $ %                   092
             IF cadflg % global in DATA % THEN *acstring* ←
             '$,*acstring*;                                         093
             litapflag ← FALSE; %to prevent bypassing rstlit in
             setlit%                                                094
           CASE nlmode OF                                           095
              = fulldisplay:                                        096
                 BEGIN                                              097
                 % indicate which accum being used and its contents%
                                                                    098
                    *mess* ← "accumulator  #";                      099
                    *mess* ← *mess*, *astrng*, ":  ", *acstring*;
```

```
                    dismes(1, $mess);
                % initialize literal area %
                    setlit();
                END;
            ENDCASE
               BEGIN
               % old TNLS bottom of loop %
                   %handle TNL hard copy feedback%
                     IF NOT nofeedbk THEN % long form %
                        BEGIN
                        typeas($blanks);
                        typeas($opstring);
                        typeas($"    ");
                        typeas($acstring);
                        END;
               END;
            %set defaults %
               proc ← $qcadd;
               *opstring* ← NULL;
               *signstr* ← '+;
               namereset ← TRUE;
            END;
      ENDCASE;
      RETURN(&result);
      END.

% ARITHMETIC %
   (xcarith) PROCEDURE
      (result,
      parsemode,
      operation,
      numptr);
      LOCAL tp2;
      REF result, operation, numptr, tp2;
      CASE parsemode OF
         =parsing:
            BEGIN
            &tp2 ← &numptr + d2sel;
            *opstring* ← numptr tp2;
            CASE operation OF
               = $add:
                  BEGIN
                  *signstr* ← '+;
                  proc ← $qcadd;
                  END;

               = $subtract:
                  BEGIN
                  *signstr* ← '-;
                  proc ← $qcsub;
                  END;
               = $multiply:
                  BEGIN
                  *signstr* ← '*;
                  proc ← $qcmult;
```

```
                END;                                                    0155
              = $divide:                                                0156
                BEGIN                                                   0157
                *signstr* ← '/;                                         0158
                proc ← $qcdiv;                                          0159
                END;                                                    0160
              ENDCASE err($"how did this happen?");                     0161
                                                                        0162
          IF nfloat($opstring, $opfloat, $opfloat + 1) THEN
          qcneg($opfloat); %convert the number to foating point,
          take care of sign%                                            0163
          qcins($opfloat,  $opstring); %insert operand in calc
          file%                                                         0164
          (proc)($accum+asub,$opfloat); %do the arithmetic and get
          answer in accum%                                              0165
          END;                                                          0166
        ENDCASE NULL;                                                   0167
      RETURN(&result);                                                  0168
      END.
                                                                        0169
% CLEAR ACCUMULATOR(S) %                                               0170
  (xclraccum) PROCEDURE (resultptr, parsemode);                         0171
      REF resultptr;                                                    0172
      CASE parsemode OF                                                 0173
        = parsing:                                                      0174
          BEGIN                                                         0175
          caclear(asub, 1);                                             0176
          CASE nlmode OF                                                0177
            = fulldisplay:                                              0178
              BEGIN                                                     0179
              dn($"0.00");                                              0180
              dpset(dspno, endfil, endfil, endfil);                     0181
              END;                                                      0182
          ENDCASE;                                                      0183
          END;                                                          0184
        ENDCASE NULL;                                                   0185
      RETURN(&resultptr);                                               0186
      END.
                                                                        0187
% CLEAR FILE %                                                         0188
  (xclrfil) PROCEDURE (resultptr, parsemode);                           0189
      REF resultptr;                                                    0190
      CASE parsemode OF                                                 0191
        = parsing:                                                      0192
          BEGIN                                                         0193
          resetf(castid.stfile); % reset pc %                           0194
          castid.stpsid ← cda.dacsp.stpsid ← orgstid;                   0195
          CASE cda.daauxiliary OF                                       0196
            = TRUE: % not showing file %                                0197
              dpset(dspno, endfil, endfil, endfil);                     0198
            ENDCASE                                                     0199
              dpset(dspyes, castid, endfil, endfil);                    0200
          END;                                                          0201
        ENDCASE NULL;                                                   0202
      RETURN(&resultptr);                                               0203
      END.
```

```
                                                                    0204
% EVALUATE EXPRESSION %                                             0205
   (xceval) PROCEDURE %Calculate value of an expression%            0206
      (result,parsemode,param);                                     0207
      LOCAL                                                         0208
         char,       %temp for parsing%                             0209
         acflag;     %true when accum value is input to 'value'
         expression%                                                0210
      LOCAL TEXT POINTER tptr, ptr1, ptr2, ptr3;  %delimiter for
      input values%                                                 0211
      REF result,param;                                             0212
      CASE parsemode OF                                             0213
         = parsing:                                                 0214
         %routine to evaluate a very simple arithmetic expression%
                                                                    0215
            BEGIN                                                   0216
            acflag ← FALSE;                                         0217
            vaccum[0] ← vaccum[1] ← 0;                              0218
            ptr1 ← param;                                           0219
            ptr1[1] ← 1;                                            0220
            ptr3 ← param [d2sel];                                   0221
            ptr3[1] ← param[d2sel+1];                               0222
            LOOP                                                    0223
               BEGIN                                                0224
               FIND ptr1 >;                                         0225
               CASE TRUE OF                                         0226
                  = (FIND ('+/'a/SP) ↑ptr1): proc ← $qcadd;         0227
                  = (FIND ('-/'s) ↑ptr1): proc ← $qcsub;            0228
                  = (FIND ('*/'x/'m) ↑ptr1): proc ← $qcmult;        0229
                  = (FIND ('//'d) ↑ptr1): proc ← $qcdiv;            0230
                  = (FIND D): proc ← $qcadd;                        0231
                  = (FIND '# ↑ptr1): acflag ← TRUE;                 0232
                  ENDCASE err($"invalid expression");               0233
               LOOP                                                 0234
                  BEGIN                                             0235
                  IF acflag THEN                                    0236
                     BEGIN                                          0237
                     IF NOT FIND ptr1 > 1$2D ↑ptr2                  0238
                     THEN err($"invalid expression");               0239
                     *tstrng* ← ptr1 ptr2;                          0240
                     char ← VALUE($tstrng)*2-2;                     0241
                     [proc]($vaccum,$accum+char);                   0242
                     acflag ← FALSE;                                0243
                     EXIT LOOP;                                     0244
                     END                                            0245
                  ELSE                                              0246
                     BEGIN                                          0247
                     IF FIND ptr1 > '# ↑ptr1 THEN                   0248
                        BEGIN                                       0249
                        acflag ← TRUE;                              0250
                        REPEAT LOOP;                                0251
                        END;                                        0252
                     FIND ptr1 > ('+/'-/TRUE) $(D/'.) ↑ptr2;        0253
                     *opstring* ← ptr1 ptr2;                        0254
                     FIND SF(*opstring*) ↑tptr;                     0255
                     IF nfloat($opstring,$opfloat, $opfloat + 1) THEN
```

```
                                                                    0256
                    qcneg($opfloat); %convert to floating point and
                    take care of sign %                             0257
                    [proc]($vaccum,$opfloat);                       0258
                    EXIT LOOP;                                      0259
                    END;                                            0260
                 END;                                               0261
              ptr1 ← ptr2;                                          0262
              ptr1[1] ← ptr2[1];                                    0263
              IF ptr2[1] >= ptr3[1] THEN EXIT LOOP;                 0264
              END;                                                  0265
           *opstring* ← NULL;                                       0266
           qfloutp($vaccum,$opstring,2);                            0267
           IF nlmode = fulldisplay THEN                             0268
              dn($opstring)                                         0269
           ELSE % typewriter %                                      0270
              typeas($opstring);                                    0271
        END;                                                        0272
     ENDCASE;                                                       0273
     RETURN(&result);                                               0274
     END.                                                           0275
(xcevend) PROCEDURE %operate on accum use temp accum, vaccum, as
operand%                                                            0276
     (result,parsemode,operation);                                  0277
     LOCAL TEXT POINTER ptr;                                        0278
     LOCAL STRING char[1];                                          0279
     REF result, operation;                                         0280
     CASE parsemode OF                                              0281
        = parsing:                                                  0282
           BEGIN                                                    0283
           CASE operation OF                                        0284
              = $add:                                               0285
                 BEGIN                                              0286
                 *signstr* ← '+;                                    0287
                 proc ← $qcadd;                                     0288
                 END;                                               0289
              = $subtract:                                          0290
                 BEGIN                                              0291
                 *signstr* ← '-;                                    0292
                 proc ← $qcsub;                                     0293
                 END;                                               0294
              = $multiply:                                          0295
                 BEGIN                                              0296
                 *signstr* ← '*;                                    0297
                 proc ← $qcmult;                                    0298
                 END;                                               0299
              = $divide:                                            0300
                 BEGIN                                              0301
                 *signstr* ← '/;                                    0302
                 proc ← $qcdiv;                                     0303
                 END;                                               0304
           ENDCASE RETURN(FALSE);                                   0305
           qcins($vaccum, $opstring); %insert value of expression in
           CALC file%                                               0306
           [proc]($accum+asub,$vaccum);                             0307
           END;                                                     0308
```

```
      ENDCASE;                                                          0309
      RETURN(&result);                                                  0310
      END.                                                              0311
% FORMAT CHANGE %                                                      0312
   (xfdigits) PROCEDURE %set number of digits after the decimal%        0313
      (result,parsemode,                                                0314
      param,   %LEFT of RIGHT of decimanal %                            0315
      value);     %number of digits %                                   0316
      LOCAL cacadflg, cafldl,char;                                      0317
      LOCAL tp2;                                                        0318
      LOCAL STRING cafstr[20];                                          0319
      REF result,param,value, tp2;                                      0320
      CASE parsemode OF                                                 0321
         = parsing:                                                     0322
          BEGIN                                                         0323
          &tp2 ← &value + d2sel;                                        0324
          *cafstr* ← value tp2;                                         0325
          char ← VALUE($cafstr);                                        0326
          CASE param OF                                                 0327
             = $right:                                                  0328
                BEGIN                                                   0329
                IF char NOT IN [0,5] THEN                               0330
                   BEGIN                                                0331
                   err($formdigerr);                                    0332
                   END                                                  0333
                ELSE                                                    0334
                   BEGIN                                                0335
                   dfoutm.fld2 ← char;                                  0336
                   dfoutm.fld1 ← 12 - char;                             0337
                   END;                                                 0338
                END;                                                    0339
             = $left:                                                   0340
                BEGIN                                                   0341
                IF char NOT IN [0,9] THEN                               0342
                   BEGIN                                                0343
                   err($formdigerr);                                    0344
                   END                                                  0345
                ELSE                                                    0346
                   BEGIN                                                0347
                   dfoutm.fld1 ← char;                                  0348
                   dfoutm.round ← dfoutm.fld1 + dfoutm.fld2;            0349
                   IF dfoutm.round > 12 THEN                            0350
                      BEGIN                                             0351
                      dfoutm ← 0640141202008;                           0352
                      err($formdigerr);                                 0353
                      RETURN;                                           0354
                      END;                                              0355
                   END;                                                 0356
                END;                                                    0357
          ENDCASE NULL;                                                 0358
         END;                                                           0359
      ENDCASE NULL;                                                     0360
      RETURN(&result);                                                  0361
      END.                                                              0362
   (xcjust) PROCEDURE %right or left justify number %                   0363
      (result,parsemode,                                                0364
```

```
    param);   %LEFT or RIGHT of decimal %                                0365
    REF result, param;                                                   0366
    CASE parsemode OF                                                    0367
       = parsing:                                                        0368
          BEGIN                                                          0369
          CASE param OF                                                  0370
             = $right:                                                   0371
                BEGIN                                                    0372
                dfoutm. JUST < 1;                                        0373
                calflg ← FALSE;                                          0374
                END;                                                     0375
             = $left:                                                    0376
                BEGIN                                                    0377
                IF cacflg THEN                                           0378
                   BEGIN                                                 0379
                   calflg ← TRUE;                                        0380
                   dfoutm.just ← 1;                                      0381
                   END                                                   0382
                ELSE                                                     0383
                   BEGIN                                                 0384
                   dfoutm.just ← 3;                                      0385
                   calflg ← FALSE;                                       0386
                   END;                                                  0387
                END;                                                     0388
          ENDCASE;                                                       0389
       END;                                                              0390
    ENDCASE;                                                             0391
    RETURN(&result);                                                     0392
    END.                                                                 0393
(xcomma) PROCEDURE (result,parsemode,param);                             0394
% set flag to insert commas in formatted number%                         0395
    REF result, param;                                                   0396
    CASE parsemode OF                                                    0397
       = parsing:                                                        0398
          BEGIN                                                          0399
          IF param = 1 THEN                                              0400
             cacflg ← TRUE                                               0401
          ELSE cacflg ← FALSE;                                           0402
          END;                                                           0403
    ENDCASE;                                                             0404
    RETURN(&result);                                                     0405
    END.                                                                 0406
(xdollar) PROCEDURE (result,parsemode,param);                            0407
% set flag to insert dollar sign in formatted number%                   0408
    REF result, param;                                                   0409
    CASE parsemode OF                                                    0410
       = parsing:                                                        0411
          BEGIN                                                          0412
          IF param = 1 THEN                                              0413
             cadflg ← TRUE                                               0414
          ELSE cadflg ← FALSE;                                           0415
          END;                                                           0416
    ENDCASE;                                                             0417
    RETURN(&result);                                                     0418
    END.                                                                 0419
(xcfeedb) PROCEDURE (result,parsemode,param);                            0420
```

```
   % set flag to abbreviate feedback in TNLS%                          0421
      REF result, param;                                               0422
      CASE parsemode OF                                                0423
         = parsing:                                                    0424
            BEGIN                                                      0425
            IF param = 1 THEN                                          0426
               nofeedbk ← TRUE                                         0427
            ELSE nofeedbk ← FALSE;                                     0428
            END;                                                       0429
      ENDCASE;                                                         0430
      RETURN(&result);                                                 0431
      END.                                                             0432
% SHOW ACCUMULATORS %                                                 0433
   (xcshoaccums) PROC (resultptr, parsemode);% show accumulators %
                                                                       0434
      %convert the accumulators to a string, separate them with the
      string "separator" followed by ordinal of accum followed by
      colon followed by a space, process "account" number of values.
      Destination string is destring.  Due to loader error in
      userprog, acc has to be passed - it is ACCUM%                    0435
      LOCAL index;                                                     0436
      LOCAL STRING acstr[30], ord[5], destring[350];                   0437
      REF resultptr;                                                   0438
                                                                       0439
      CASE parsemode OF                                                0440
         = parsing:                                                    0441
            BEGIN                                                      0442
            index ← 0;                                                 0443
            *destring* ← NULL;                                         0444
            DO                                                         0445
               BEGIN                                                   0446
               qfloutp($accum + index, $acstr, 2); %convert%           0447
               *ord* ← STRING((index+2)/2), ": ";                      0448
               *destring* ← *destring*, EOL, "  ", *ord*, *acstr*;
                                                                       0449
               END                                                     0450
            UNTIL (index ← index + 2) > 19;                            0451
            fbctl(typecalit, $destring);                               0452
            dpset(dspno, endfil, endfil, endfil);                      0453
            END;                                                       0454
         ENDCASE NULL;                                                 0455
      RETURN(&resultptr);                                              0456
      END.
                                                                       0457
% SHOW FILE %                                                         0458
   (xcshofil) PROCEDURE (resultptr, parsemode);                        0459
      LOCAL da, csp, spot;                                             0460
      REF resultptr, da;                                               0461
      CASE parsemode OF                                                0462
         = parsing:                                                    0463
            BEGIN                                                      0464
            % get rid of old da %                                      0465
               &da ← lda(); % address of da being split %              0466
            % split screen %                                           0467
               spot ← da.daleft + (23 * da.dainc);                     0468
               <DSPGEN, clearda> (&da);                                0469
```

```
            <DSPGEN, clrall> (&da, TRUE); %zero out LSRT entries%
                                                                    0470
            %call vertical split to put old file on right side%
                                                                    0471
               IF NOT vsplit(&da, spot, spot + da.dahinc) THEN
                                                                    0472
                  BEGIN                                             0473
                  fbctl(typecalit, $spliterr); % bad split %        0474
                  RETURN(&resultptr);                               0475
                  END;                                              0476
               csp ← cda.dacsp;                                     0477
               delda(&cda);                                         0478
               &cda ← &da;                                          0479
               cda.daempty ← FALSE;                                 0480
               cda.dacsp ← csp;                                     0481
            % recreate display %                                    0482
               <DSPGEN, alldsp> ();                                 0483
            END;                                                    0484
         ENDCASE NULL;                                              0485
      RETURN(&resultptr);                                           0486
      END.

                                                                    0487
% TOTAL %                                                          0488
   (xctotl) PROCEDURE (resultptr, parsemode);                       0489
      REF resultptr;                                                0490
      CASE parsemode OF                                             0491
         = parsing:                                                 0492
            BEGIN                                                   0493
            *signstr* ← 'T;                                         0494
            qcins($accum + asub, $opstring);                        0495
            dpset(dspno, endfil, endfil, endfil);                   0496
            CASE nlmode OF                                          0497
               = typewriter: typeas($opstring);                     0498
            ENDCASE;                                                0499
            *signstr* ← '+;                                         0500
            END;                                                    0501
         ENDCASE NULL;                                              0502
      RETURN(&resultptr);                                           0503
      END.
                                                                    0504
% USE ACCUMULATOR # %                                              0505
   (xcuseaccum) PROCEDURE(resultptr, parsemode, numptr);            0506
      LOCAL index;                                                  0507
      LOCAL TEXT POINTER tp2;                                       0508
      REF resultptr, numptr, tp2;                                   0509
      CASE parsemode OF                                             0510
         = parsing:                                                 0511
            BEGIN                                                   0512
            &tp2 ← &numptr + d2sel;                                 0513
            *astrng* ← numptr tp2; % string containing number %     0514
            IF (index ← VALUE($astrng) * 2 - 2) NOT IN [0,19] THEN
                                                                    0515
               fbctl(typelit, $badaccno)                            0516
            ELSE                                                    0517
               % update the master window !! %                      0518
```

```
            asub ← index; % change main subscript %                    0519
         END;                                                          0520
      ENDCASE NULL;                                                    0521
   dpset(dspno, endfil, endfil, endfil);                               0522
   RETURN(&resultptr);                                                 0523
   END.
                                                                       0524
% USE SAVED ACCUMULATORS %                                            0525
  (xcusesaved) PROCEDURE(resultptr, parsemode);                        0979
   LOCAL stid, index;                                                  0980
   LOCAL STRING                                                        0981
      temstr[4],                                                       0982
      calcstr[4],     % special calc signature %                       0983
      sigstr[35],    % for checking signature %                        0984
      value[22];                                                       0985
   LOCAL TEXT POINTER start, end;
                                                                       0986
   REF resultptr;                                                      0987
   CASE parsemode OF                                                   0988
      = parsing:                                                       0989
         BEGIN                                                         0990
         %initialize locals%                                           0991
            *calcstr* ← "CALC";                                        0992
            index ← 0;                                                 0993
            stid ← orgstid;                                            0994
            stid.stfile ← cda.dacsp.stfile;                            0995
            stid ← <FILMNP, getsub>(stid); %location of info%          0996
            fechsig(stid, $sigstr);                                    0997
            *temstr* ← *sigstr*[1 TO 4];                               0998
            CCPOS SF(stid);                                            0999
            IF *temstr* # *calcstr* THEN                               01000
               BEGIN                                                   01001
               csysbad();                                              01002
               RETURN(&resultptr);                                     01003
               END;                                                    01004
         %check for nothing saved%                                     01005
            IF NOT FIND ["CALC ACCUMS:"] ↑start THEN                   01006
               BEGIN                                                   01007
               IF NOT FIND start ['//] <CH ↑end THEN                   01008
                  BEGIN                                                01009
                  csysbad();                                           01010
                  RETURN(&resultptr);                                  01011
                  END;                                                 01012
               END;                                                    01013
            %get and convert values%                                   01014
            DO                                                         01015
               BEGIN                                                   01016
               IF NOT FIND start ['//] <CH ↑end THEN                   01017
                  BEGIN                                                01018
                  csysbad();                                           01019
                  RETURN(&resultptr);                                  01020
                  END;                                                 01021
               *value* ← start end;                                    01022
               IF nfloat($value, $accum + index, $accum + index +
               1) THEN qcneg($accum+index);                            01023
               FIND end >CH ↑start; %get over slash delimiter%
```

```
                                                                    01024
      END                                                           01025
   UNTIL (index ← index + 2) > 19;                                  01026
%retrieve format variables%                                         01027
   stid ← <FILMNP, getsuc>(stid); %location of format
   flags%                                                           01028
   *sigstr* ← NULL:                                                 01029
   fechsig(stid, $sigstr);                                          01030
   *temstr* ← *sigstr*[1 TO 4];                                      01031
   IF *temstr* #* *calcstr* THEN                                    01032
      BEGIN                                                         01033
      csysbad();                                                    01034
      RETURN(&resultptr);                                           01035
      END;                                                          01036
   CCPOS SF(stid);                                                  01037
   IF NOT FIND ["FORMAT:"] ↑start THEN                              01038
      BEGIN                                                         01039
      csysbad();                                                    01040
      RETURN(&resultptr);                                           01041
      END;                                                          01042
   IF NOT FIND  3D ↑end THEN                                        01043
      BEGIN                                                         01044
      csysbad();                                                    01045
      RETURN(&resultptr);                                           01046
      END;                                                          01047
   *value* ← start end;                                             01048
   cacflg ← IF *value*[1] = '0 THEN FALSE ELSE TRUE;
                                                                    01049
   cadflg ← IF *value*[2] = '0 THEN FALSE ELSE TRUE;
                                                                    01050
   calflg ← IF *value*[3] = '0 THEN FALSE ELSE TRUE;
                                                                    01051
   %retrieve format mask%                                           01052
      stid ← <FILMNP, getsuc>(stid); %location of format
      mask%                                                         01053
      *sigstr* ← NULL;                                              01054
      fechsig(stid, $sigstr);                                       01055
      *temstr* ← *sigstr*[1 TO 4];                                  01056
      IF *temstr* # *calcstr* THEN                                  01057
         BEGIN                                                      01058
         csysbad();                                                 01059
         RETURN(&resultptr);                                        01060
         END;                                                       01061
      CCPOS SF(stid);                                               01062
      FIND SF(stid) ↑start;                                         01063
      IF NOT FIND 1$12D ↑end THEN                                   01064
         BEGIN                                                      01065
         csysbad();                                                 01066
         RETURN(&resultptr);                                        01067
         END;                                                       01068
      *value* ← start end;                                          01069
      dfoutm ← VALUE($value);                                       01070
      %code to verify format may be inserted here - if
      user has messed with this statement, he may get a
      sudden illegal instruction eventually%                        01071
      dpset(dspno, endfil, endfil, endfil);                         01072
```

```
                END;                                                    01073
              ENDCASE NULL;                                             01074
            RETURN(&resultptr);                                         01075
            END.
                                                                        01076
   (csysbad) PROCEDURE;                                                  0622
      REF resultptr;                                                     0623
      fbctl(typelit, $acsaverr);                                         0624
      RETURN;                                                            0625
      END.                                                               0626
                                                                         0627
% WRITE NEW FILE %                                                      0628
   (xcwritef) PROCEDURE (resultptr, parsemode, fnamptr); %calculator
   write file%                                                           0629
      %allow user to update the calc-ident file to a new file in his
      directory.%                                                        0630
                                                                         0631
      LOCAL STRING oldnam[50], relfilname[200];                          0632
      REF fnamptr, resultptr;                                            0633
      LOCAL nameaddress;                                                 0634
                                                                         0635
      CASE parsemode OF                                                  0636
         = parsing:                                                      0637
            BEGIN                                                        0638
            % move file name to local string %                           0639
               CASE lnbfls( &fnamptr, 0, $relfilname) OF                 0640
                  = lhostn: NULL;                                        0641
                  ENDCASE                                                0642
                     err($"Remote File Manipulations Not Implemented
                     Yet");                                              0643
            nameaddress ← fnamptr.RH;                                    0644
            %do the actual update function%                              0645
               updtfl(castid.stfile, newversion, $relfilname);           0646
               dismes(2, [flntadr(castid.stfile)].flastr);               0647
                  %calling routine will get freflnt to close the file
                  updated to%                                            0648
            %re-open the old base file%                                  0649
               clcname($oldnam); %set calc-ident file name%              0650
               castid ← orgstid;                                         0651
               IF NOT (castid.stfile ← <CORENL,
               cloafil>($oldnam))THEN                                    0652
                  abortsubsystem($caupderr);                             0653
               cda.dacsp ← castid;                                       0654
            dpset(dspno, endfil, endfil, endfil);                        0655
            END;                                                         0656
         ENDCASE NULL;                                                   0657
      RETURN(&resultptr);                                                0658
      END.
                                                                         0659
% INSERT/REPLACE %                                                      0660
   (xcinsac) PROCEDURE                                                   0661
      (result,parsemode);  %Insert or replace %                          0662
      REF result;                                                        0663
      CASE parsemode OF                                                  0664
         = parsing:                                                      0665
            BEGIN                                                        0666
```

```
            result ← result[d2sel] ← $acstring;                          0667
            result[d2sel].stastr ← result.stastr +1;                     0668
            result[1] ← 1;                                               0669
            result[d2sel+1] ← acstring.L + 1;                            0670
            END;                                                         0671
         ENDCASE;                                                        0672
      RETURN (&result);                                                  0673
      END.
                                                                         0674
% RE-ENTER SUBSYSTEM CODE %                                             0675
   (xcreenter) PROCEDURE (resultptr, parsemode); % calc reenter %
                                                                         0676
      REF resultptr;                                                     0677
      LOCAL TEXT POINTER                                                 0678
         tp1, tp2, h1, h2, u1, u2, f1, f2, n1, n2, v1, v2;               0679
      LOCAL STRING                                                       0680
         oldname[30], % calc-ident string only %                         0681
         nowname[70]; % whole name, currently in cda %                   0682
      CASE parsemode OF                                                  0683
         =parsing:                                                       0684
            BEGIN                                                        0685
            IF cda.daexis % our global still a da %                      0686
            AND NOT cda.daauxiliary % being viewed %                     0687
            THEN                                                         0688
               BEGIN                                                     0689
               clcname($oldname);                                        0690
               filnam(cda.dacsp.stfile, $nowname);                       0691
               FIND SF(*nowname*) ↑tp1;                                  0692
               lnbfls( $tp1, 0, $nowname);                               0693
               IF NOT FIND tp1 > ['.] < CH ↑f2 THEN                      0694
                  err($"system string error");                           0695
               *nowname* ← tp1 f2;                                       0696
               *nowname* ← *nowname*[1 TO oldname.L];                    0697
               IF *nowname* # *oldname* % he loaded another file
               there% THEN                                               0698
                  cafilinitialize() % punt %                             0699
               ELSE                                                      0700
                  BEGIN % good da, right file %                          0701
                  getail(castid); % he may have added something - we
                  were in this plex before %                             0702
                  dpset(dspyes, castid, endfil, endfil);%recreate%
                                                                         0703
                  END;                                                   0704
               END                                                       0705
            ELSE IF NOT cda.daexis THEN cafilinitialize() % punt %
                                                                         0706
            END;                                                         0707
         ENDCASE NULL;                                                   0708
      RETURN END.

                                                                         0709
% TERMINATION CODE %                                                    0710
   (xcquit) PROCEDURE (resultptr, parsemode);                            0711
      % calculator TERMINATION CODE %
                                                                         0712
      LOCAL stid,   %cal-file origin%                                    0713
```

```
    account,     % number of words of information to save % 0714
    savsig,  % temp for NLS signature value %               0715
    trapping,    % TRUE if we have disarmed control C %     0965
    capsav,    % save capabilities while trapping so they can be
    restored when contol c is rearmed %                     0966
    index;     %accumulator save array index%               0716
LOCAL STRING                                                0717
    save[250] , %enough for 10 accums%                      0718
    errsr[150], % for error message from coropnfil %        0719
    calcsig[5],                                             0720
    temp[20];                                               0721
LOCAL TEXT POINTER start, finish;                           0722
REF resultptr;
                                                            0723
CASE parsemode OF                                           0724
    =parsing:                                               0725
        BEGIN                                               0726
        trapping ← FALSE;                                   0967
        namereset ← FALSE;                                  0727
        cpadent ← FALSE;                                    0728
        % save state %                                      0729
        *calcsig* ← "CALC";                                 0730
        savsig ← cinit; % standard NLS signature %          0731
        ON SIGNAL ELSE                                      0968
            BEGIN                                           0969
            % reset ident string %                          0970
                cinit ← savsig;                             0971
            % re-arm control C if necessary. %              0972
                IF trapping := FALSE THEN notrapcc(capsav); 0973
            END;                                            0974
        trapping ← TRUE;                                    0975
        capsav ← trapcc();                                  0976
        cinit ← setcinit($calcsig);                         0732
        account ← 19; %number of words - 1%                 0733
        *save* ← "CALC ACCUMS:";                            0734
        index ← 0;                                          0735
        %convert accumulators to character string%          0736
            DO                                              0737
                BEGIN                                       0738
                qfloutp($accum + index, $temp,2);           0739
                *save* ← *save*, *temp*, '/ ';              0740
                END                                         0741
            UNTIL (index ← index + 2) > account;            0742
            *save* ← *save*, '; ';                          0743
            stid ← cda.dacsp;                               0744
            stid.stpsid ← orgstid;                          0745
        %store accumulators in user file%                   0746
            start ← finish ← $save;                         0747
            start.stastr ← 1;                               0748
            finish.stastr ← 1;                              0749
            start[1] ← 1;                                   0750
            finish[1] ← save.L + 1;                         0751
            stid ← cinssta(stid, sucdir, $start, $finish);  0752
            *save* ← "FORMAT:";                             0753
            *save* ← *save*, STRING(cacflg);                0754
            *save* ← *save*, STRING(cadflg);                0755
```

```
                *save* ← *save*, STRING(calflg);                    0756
             %store format information in user file%                0757
                finish[1] ← save.L + 1;                             0758
                stid ← cinssta(stid, sucdir, $start, $finish);      0759
                *save* ← STRING(dfoutm);                            0760
             %store format mask in user file%                       0761
                finish[1] ← save.L + 1;                             0762
                stid ← cinssta(stid, sucdir, $start, $finish);      0763
             cinit ← savsig;                                        0764
             IF trapping := FALSE THEN notrapcc(capsav);            0977
             CASE cda.daauxiliary OF                                0765
                = TRUE: % TNLS or DNLS with no displayed file %     0766
                   delda(&cda); % file will get closed with next
                   recreate display %                               0767
                ENDCASE                                             0768
                   BEGIN                                            0769
                   clearda(0); %erase entire text area%             0770
                   clrall(0, TRUE); %erase all line seg ref tables%
                                                                    0771
                   fixbnd(TRUE, cda.daright, cda.left, TRUE);       0772
                   fixbuf();                                        0978
                   dpset(dspallf, endfil, endfil, endfil);          0773
                   END;                                             0774
             END;                                                   0775
          ENDCASE dismes(0);; % for error, cleanup %                0778
       RETURN (&resultptr);                                         0779
       END.
                                                                    0780
% INITIALIZATION %
                                                                    0781
   (cafilinitialize) PROCEDURE; % calc file set up %                0782
      LOCAL fileno;                                                 0783
      LOCAL STRING flnam[50];                                       0784
      dpset(dspno,endfil,endfil,endfil): % set globals = no recreate %
                                                                    0785
      % get auxiliary da %                                         0786
         &cda ← newda(); %get new da%                               0787
         intdafl(&cda); %initialize newda%                          0788
         cda.daauxiliary ← TRUE; % use this flag to tell whether to
         bother with screen %                                       0789
      %load or initialize CALC-IDENT file%                          0790
         cda.dacsp ← orgstid;                                       0791
         IF NOT cacfile (:fileno) THEN                              0792
            abortsubsystem($badcfile);                              0793
         cda.dacsp.stfile ← fileno;                                 0794
      %initialize da to display file%                               0795
         castid ← cda.dacsp ← <STRMNP, getail> (<FILMNP, getsub>    0796
            (cda.dacsp));                                           0797
         IF cda.dacsp.stpsid = origin %empty file% THEN             0798
            BEGIN                                                   0799
            clcname($flnam);                                        0800
            updtfl(fileno, FALSE, $flnam); %don't care if update
            actually happened or not%                               0801
            END;                                                    0802
      RETURN END.
                                                                    0803
```

```
% CLEAR ACCUMULATOR(S) %                                                 0804
   (caclear) PROCEDURE (accx,clrwhat); %clear accum pointed at by accx
   or all accums%                                                        0805
      IF clrwhat = 1 THEN                                                0806
         BEGIN                                                           0807
         accum[accx] ← accum[accx+1] ← 0;                                0808
         RETURN;                                                         0809
         END                                                             0810
      ELSE                                                               0811
         BEGIN                                                           0812
         accx ← 0;                                                       0813
         DO                                                              0814
            accum[accx] ← accum[accx+1] ← 0                              0815
         UNTIL (accx ← accx +2) >= 19;                                   0816
         END;                                                            0817
      RETURN;                                                            0818
      END.
                                                                         0819
% SUPPORT ROUTINES %                                                     0820
   (errx) PROCEDURE (whence,ar); %JSYS dfout has returned with an error
   condidtion. R4 contains error mnemonics. FLOTX1 and  FLOTX2 indicate
   column overflow. All other error returns are either calulator or
   tenex bugs. If the error occurred on an accumulator and the format is
   the default, maximum no. of digits, the accum will we set to zero.
   If the format is for less than the maximum, the format will be
   changed to the maximum.%                                              0821
      REF ar;                                                            0822
      IF whence = 1 THEN err($insizerr)                                  0823
      ELSE                                                               0824
         BEGIN                                                           0825
         IF dfoutm.round < 12 THEN                                       0826
            BEGIN                                                        0827
            dfoutm←064014120200B;                                        0828
            err($acsizerr);                                              0829
            END                                                          0830
         ELSE                                                            0831
            BEGIN                                                        0832
            ar ← ar[1] ← 0;                                              0833
            err($acsizdef);                                              0834
            END;                                                         0835
         END;                                                            0836
      RETURN;                                                            0837
      END.

                                                                         0838
   (cacfile) PROCEDURE; %get/create calc file%                           0839
      %If user has done a Quit Return, take care of his file, which had
      the partial copy closed to make it read only.  Otherwise, load his
      calc-ident file, if he has one.  If not, create a null calc-ident
      file%                                                              0840
      LOCAL fileno,  %file number%                                       0841
            stid;  %origin of file if Quit Return was done%              0842
      LOCAL STRING flnam[50],filestr[65];                                0843
      ON SIGNAL                                                          0844
         =-5: %bad file%                                                 0845
            RETURN(FALSE, fileno);%let calling routine handle it%        0846
```

```
   ELSE NULL;                                                          0847
   clcname($flnam); %get calc-ident file name%                         0848
   IF NOT (fileno ← opwk(0, $flnam)) THEN RETURN(FALSE, FALSE);        0849
   RETURN(TRUE,fileno);                                                0850
   END.
                                                                       0851
(clcname) PROCEDURE(flnam);%set calculator file name to login
directory CALC file%                                                   0852
   REF flnam;                                                          0853
   !JSYS gjinf; %get login directory number%                           0854
   gdname(r1,&flnam); %conver to string%                               0855
   *flnam* ← '<,*flnam*,">CALC-",*initsr*;                             0856
   RETURN;                                                             0857
   END.                                                                0858
                                                                       0859
(qcins) PROCEDURE (cfloat,opstrng); %insert value%                     0860
   %into calc-ident file - updates global CASTID %                     0861
   %cfloat = value to insert at stid, csign is sign and/or original
   operator. opstrng is address of final formatted operand.%           0862
   LOCAL tempstid;                                                     0863
   LOCAL TEXT POINTER tp1, tp2;                                        0864
   REF cda, cfloat,  opstrng;
                                                                       0865
   qfloutp(&cfloat,&opstrng,1);                                        0866
   *opstrng* ← *opstrng*, SP, *signstr*;                               0867
   tempstid ← castid;                                                  0868
   tp1 ← tp2 ← $opstrng;                                               0869
   tp1.stastr ← 1;                                                     0870
   tp2.stastr ← 1;                                                     0871
   tp1[1] ← 1;                                                         0872
   tp2[1] ← opstrng.L + 1;                                             0873
   castid ← cinssta(tempstid,sucdir,$tp1,$tp2);                        0874
   %recreate display and take care of scrolling%                       0875
   IF nlmode = fulldisplay AND NOT cda.daauxiliary THEN                0876
      BEGIN                                                            0877
      IF cda.dacrow < cda.damrow THEN                                  0878
         BEGIN                                                         0879
         dpset(dspstrc,tempstid,endfil,endfil);                        0880
         seldsp();                                                     0881
         END                                                           0882
      ELSE                                                             0883
         BEGIN                                                         0884
         cda.dacsp ← castid;                                           0885
         dafrmt(&cda, 0);                                              0886
         END;                                                          0887
      END;                                                             0888
   IF NOT cda.daauxiliary THEN bmoff();                                0889
   RETURN;                                                             0890
   END.
                                                                       0891
(qfloutp) PROC(ar, os,whence);%qfloating output a value at address ar
to string at address os%                                               0892
   LOCAL TEXT POINTER tp1, tp2, tp3;                                   0893
   LOCAL i, j, k, fst;                                                 0894
   LOCAL STRING cos[16];                                               0895
   REF os;                                                             0896
```

```
*cos* ← "                    ";                                          0697
!MOVE r3,ar;                                                            0698
!MOVE r2,0(r3);                                                         0699
!MOVE r3,1(r3);                                                         0700
r4←dfoutm; %indicator of precision%                                     0701
r1 ← &os + 440700000001B; %TENEX string designator constant%            0702
IF NOT SKIP !JSYS dfout THEN errx(whence,ar);                           0703
os.L ← os.M;                                                            0704
CCPOS SF(*os*);                                                         0705
FIND ↑tp1 S( D/ './ 'E/ '-) ↑tp2;                                       0706
*os* ← tp1 tp2;                                                         0707
IF *os*[1] = '- THEN fst ← 2                                            0708
ELSE fst ← 1;                                                           0709
FOR i ← fst UP UNTIL > os.L DO                                          0710
   IF *os*[i] = '0 THEN *os*[i] ← SP                                    0711
   ELSE EXIT;                                                           0712
IF cacflg THEN                                                          0713
   BEGIN                                                                0714
   j ← os.L + 2;                                                        0715
   FOR i ← os.L DOWN UNTIL = 1 DO                                       0716
      IF *os*[i] # '. THEN                                              0717
         BEGIN                                                          0718
         *cos*[j] ← *os*[i];                                            0719
         j ← j-1;                                                       0720
         END                                                            0721
   ELSE EXIT;                                                           0722
   *cos*[j] ← '.;                                                       0723
   i← i-1;                                                              0724
   j ← j-1;                                                             0725
   LOOP BEGIN                                                           0726
   k ← i;                                                               0727
   FOR i DOWN UNTIL = k-3 DO                                            0728
      BEGIN                                                             0729
      IF i < 1 THEN EXIT LOOP 2;                                        0730
      IF *os*[i] = SP THEN EXIT LOOP 2;                                 0731
      IF *os*[i] = '- THEN EXIT LOOP 2;                                 0732
      *cos*[j] ← *os*[i];                                               0733
      j ← j-1;                                                          0734
      END;                                                              0735
   IF *os*[i] = SP THEN EXIT LOOP;                                      0736
   IF i <= 1 THEN EXIT LOOP;                                            0737
   IF *os*[i] # '- THEN                                                 0738
      BEGIN                                                             0739
      *cos*[j] ← ',;                                                    0740
      j← j-1;                                                           0741
      REPEAT LOOP;                                                      0742
      END                                                               0743
   ELSE EXIT LOOP;                                                      0744
   END;                                                                 0745
   IF calflg THEN BEGIN                                                 0746
      FIND SF(*cos*) SSP (D/'.) ↑tp1 ←tp1;                              0747
      FIND SE(*cos*) $NP ↑tp2;                                          0748
      IF *os*[1] = '- THEN k ← 2                                        0749
      ELSE k ← 1;                                                       0750
      *os*[k TO os.M] ← tp1 tp2;                                        0751
      END                                                               0752
```

```
        ELSE                                                    0953
           BEGIN                                                0954
           FIND SE(*cos*) ↑tp3 $NP ↑tp2;                        0955
           ST tp2 tp3 ← NULL;                                   0956
           IF *os*[1] = '- THEN k ← 2                           0957
           ELSE k← 1;                                           0958
           *os*[k TO os.M] ← *cos*;                             0959
           END;                                                 0960
        END;                                                    0961
     IF *os*[os.L] = '. THEN os.L ← os.L-1;                     0962
     RETURN END.
                                                                0963
FINISH of PSCALC                                                0964
```

PSHELP

(MLK)PSHELP (MLK)PSHELP (MLK)PSHELP (MLK)PSHELP (MLK)PSHELP (M
(MLK)PSHELP (MLK)PSHELP (MLK)PSHELP (MLK)PSHELP (MLK)PSHELP (M

```
< NLS, PSHELP.NLS;19, >, 21-OCT-74 10:02 HGL ;;;;
FILE pshelp % L10 <rel-nls>pshelp %% (110,) (rel-nls,pshelp.rel,) % 0252
% DECLARATIONS %                                                  0254
   REF inpt;                                                     01298
   REF conrng, qda, qsw, curstk, qstorblk, qnewstmt, hlpcmdstk; % For
   help system %                                                 0366
% HELP Parsefunctions %                                          02
   (lookback) PROCEDURE( % parsing function which looks at the next
   input char.  If it is an "←" character, then it returns TRUE,
   otherwise it returns FALSE %                                  0864
      % FORMAL ARGUMENTS %                                       0865
         resultptr,     % ptr to the result record %             0866
         parsemode,     % parsing mode %                         0867
         string);       % ptr to help string %                   0868
      REF resultptr, string;                                     0869
      % -------------- %                                        0870
         CASE parsemode OF                                       0871
            = parsing:                                           0872
               CASE lookc() OF                                   0873
                  = '←;                                          0874
                     inpt();                                     0875
                  ENDCASE RETURN (FALSE);                        0876
            = parsehelp:                                         0877
               *string* ← "T:/←";                                0878
            = parseqmark:                                        0879
               BEGIN                                             0880
               *string* ← NULL;                                  0881
               RETURN;                                           0882
               END;                                              0883
            ENDCASE;                                             0884
         RETURN (&resultptr );                                   0885
         END.
                                                                 0886
   (lookrpt) PROCEDURE( % parsing function which looks at the next
   input char.  If it is an REPEAT character, then it reads the char
   and returns TRUE, otherwise it returns FALSE %                0887
      % FORMAL ARGUMENTS %                                       0888
         resultptr,     % ptr to the result record %             0889
         parsemode,     % parsing mode %                         0890
         string);       % ptr to help string %                   0891
      REF resultptr, string;                                     0892
      % -------------- %                                        0893
         CASE parsemode OF                                       0894
            = parsing:                                           0895
               CASE lookc() OF                                   0896
                  = rptchar:                                     0897
                     NULL;                                       0898
                  ENDCASE RETURN (FALSE);                        0899
            = parsehelp:                                         0900
               *string* ← "RPT:";                                0901
            = parseqmark:                                        0902
               BEGIN                                             0903
               *string* ← "REPEAT";                              0904
               RETURN;                                           0905
               END;                                              0906
            ENDCASE;                                             0907
```

```
    RETURN (&resultptr );                                          0908
    END.
                                                                   0909
(mylookbug) PROC(curptr, parsemode, string);                       0910
  % lookoug looks at the next inpt character, if it is a CA, then a
  true return is taken else FALSE is returned %                    0911
  REF curptr, string;                                              0912
  %------------%                                                   0913
  CASE parsemode OF                                                0914
    = parsing:                                                     0915
      CASE lookc() OF                                              0916
        = cachar, = rptchar, = inschar:                            0917
          NULL;                                                    0918
      ENDCASE RETURN (FALSE);                                      0919
    = parsehelp:                                                   0920
      IF nlmode = fulldisplay THEN *string* ← "B:"                 0921
      ELSE *string* ← "OK:";                                       0922
    = parseqmark:                                                  0923
      BEGIN                                                        0924
      IF nlmode = fulldisplay THEN *string* ← "BUG"                0925
      ELSE *string* ← "OK";                                        0926
      RETURN;                                                      0927
      END;                                                         0928
  ENDCASE;                                                         0929
  RETURN (&curptr);                                                0930
  END.
                                                                   0931
                                                                   0932
(myrdconfirm) PROC(curptr, parsemode, string);                     01472
  % readconfirm looks at the next inpt character, if it is a
  CA/REPEAT/INSERT, then it is read and a true return is taken else
  FALSE is returned %                                              01473
  REF curptr, string;                                              01474
  %------------%                                                   01475
  CASE parsemode OF                                                01476
    = parsing:                                                     01477
      CASE lookc() OF                                              01478
        = cachar, = rptchar, = inschar:                            01479
          BEGIN                                                    01480
          % stick ptr to castring into result record %             01481
            curptr ← $castr;                                       01482
          % read over the CA %                                     01483
            inpt();                                                01484
          END;                                                     01485
      ENDCASE RETURN (FALSE);                                      01486
    = parsehelp:                                                   01487
      *string* ← "OK/T:";                                          01488
    = parseqmark:                                                  01489
      BEGIN                                                        01490
      *string* ← "OK or node";                                     01491
      RETURN;                                                      01492
      END;                                                         01493
  ENDCASE;                                                         01494
  RETURN (&curptr);                                                01495
  END.
                                                                   01496
```

```
                                                                    01497
% Help execution functions. %                                        0933
  (hlpinit) PROC (resultptr, parsemode, entrymode, intparm);        01301
     % This procedure does the initialization required upon entry into
     the help command.  If the entrymode is CNTLQ, thena control-Q was
     typed to get in:  this implies that the command data structure
     has been set up and may be followed to find tthe first item to be
     displayed.  If the entry mode  is HLPCOM, then we must set up the
     array topoint to the current subsystem.  If the HELP command has
     not been used before at this session, we must open files and
     allocate appropriate storage.  If it has been entered, the file
     will be around (check to be sure.)  In cleanup or backup modes we
     must release storage and sequences and set up for the next entry.
      %                                                             01302
     LOCAL                                                          01303
        entryptr, % pointer to subsystem stack; used to get pointer to
        name of subsystem %                                         01304
        helpad, % work space %                                      01305
        typead, % work space %                                      01306
        shortcut,                                                   01307
        entstd,                                                     01308
        topstd,                                                     01309
        end,                                                        01310
        da;                                                         01311
     LOCAL TEXT POINTER filptr;                                     01312
     LOCAL STRING namstr[100];                                      01313
     REF resultptr, entrymode, entryptr, da, intparm;               01314
     CASE parsemode OF                                              01315
        = parsing:                                                  01316
           BEGIN                                                    01317
           inhelp ← qagain := 0;  % qagain set ONLY here and in
           helphlp.  Necessary because inhelp is reset upon
           termination which happens when control q is hit again! %
                                                                    01318
           IF inhelp THEN                                           01319
              BEGIN                                                 01320
              % We are in the help command already.  Simply show the
              help branch and continue. %                           01321
              helpad ← $"help";                                     01322
              typead ← $name;                                       01323
              RETURN( helpshow(&resultptr, parsing, $typead,
              $helpad));                                            01324
              END;                                                  01325
           shortcut ← FALSE;                                        01326
           % set up destination array for this subsystem.  It has
           already been done for cntlq mode. %                      01327
              IF entrymode = $hlpcom THEN                           01328
                 BEGIN                                              01329
                 % Get storage for name stack %                     01330
                    IF NOT &hlpcmdstk THEN &hlpcmdstk ←
                    getarray(hpcmdmax + 1, $dspblk);                01331
                 hlpcmdstk ← 1;                                     01332
                 IF NOT intparm THEN                                01333
                    BEGIN                                           01334
                    % No initial parameter has been specified %     01335
                    % set up ptr to subsystem name %               01336
```

```
            &entryptr ← $sbstack + sbstkx - $sbentsize;
                                                                    01337
         % set hlpcmdstk to point to subsystem name %               01338
            hlpcmdstk[1] ← entryptr.sbnptr;                         01339
         END                                                        01340
      ELSE                                                          01341
         BEGIN                                                      01342
         shortcut ← TRUE;                                           01343
         % set hlpcmdstk to point to base %                         01344
            hlpcmdstk[1] ← $"base";                                 01345
         END;                                                       01346
      END;                                                          01347
ON SIGNAL ELSE                                                      01348
   BEGIN                                                            01349
   ON SIGNAL ELSE;                                                  01350
   abortsubsystem($"Help command system error:
   Call ARC");                                                      01351
   END;                                                             01352
% Check if help DB file is open; open if neccessary %               01353
   IF NOT hlpfileno THEN                                            01354
      BEGIN                                                         01355
      % Must open help or xhelp data base file. %                   01356
      IF hdebug THEN                                                01357
         *namstr* ← "<DOCUMENTATION>XHELP.NLS;"                     01358
      ELSE *namstr* ← "<DOCUMENTATION>HELP.NLS;";                   01359
      IF NOT (hlpfileno ← cloafil($namstr)) THEN                    01360
         abortsubsystem($"Help data base not online.  Call
         ARC");                                                     01361
      % Set bit so file is never closed %                          01362
         [flntadr(hlpfileno)].flnoclos ← TRUE;                      01363
      END;                                                          01364
% General initialization %                                         01365
   &qda ← &qsw ← &qstorblk ← &qnewstmt ← 0;                         01366
   % Redirect control-Q for use in help. %                        01367
      chntab[24] ← $helphlp .V 1B6;                                 01368
   % Set flag saying we're in Help  command. %                    01369
      inhelp ← TRUE;                                                01370
   % Initialize global error flag %                               01371
      hseqer ← FALSE;                                               01372
   % Save old da's (and jump stacks) and get a new one for
   use by query; upon quitting, these will be restored %
                                                                    01373
      IF nlmode = fulldisplay THEN                                  01374
         BEGIN                                                      01375
         clearda(0);                                                01376
         clrall( 0, TRUE );                                         01377
         % Save away auxiliary bit for calculator display
         area. %                                                    01378
            calcaux ← cda.daauxiliary ;                             01379
         % set bit in da's so that they are ignored. %             01380
            end ← (&da ← $dpyarea) + dacnt*dal;                     01381
            DO da.daauxiliary ← TRUE                                01382
            UNTIL (&da ← &da + dal) >= end;                         01383
         END;                                                       01384
      &qda ← newda();                                               01385
      intdafl(&qda);                                                01386
```

```
   IF nlmode = fulldisplay THEN                                    01387
      BEGIN                                                        01388
      alocda(&qda);                                                01389
      dpset(dspno, endfil, endfil, endfil);                        01390
      END;                                                         01391
% The sequence generator for query/help. %                        01392
   qda.dafrzl ← 0;                                                 01393
   qda.davspec ← defvs1;                                           01394
   qda.davspc2 ← defvs2;                                           01395
   qda.dausqcod ← $queryseq;                                       01396
   qda.davspec.vsusqf ← TRUE;                                      01397
   qda.davspec.vsindf ← FALSE;                                     01398
   qda.davspec.vsbrcf ← TRUE;                                      01399
   qdavspc ← qda.davspec;                                          01400
   qdavs2 ← qda.davspc2;                                           01401
% Initialize stid for first search; the TP will be
changed by lookup! %                                               01402
   filptr ← orgstid;                                               01403
   filptr.stfile ← hlpfileno;                                      01404
   filptr[1] ← 1;                                                  01405
% get stid of first branch to be shown; release space
for hlpcmdstk %                                                    01406
   IF (entstd ← hlpstrt($filptr, &hlpcmdstk)) = endfil
   THEN abortsubsystem($"Database problem. Call ARC.");
                                                                   01407
   freestring(&hlpcmdstk := 0, $dspblk);                           01408
% Get the stids of the top and the entry point-- %                01409
   topstd ← entstd;                                                01410
   topstd.stpsid ← orgstid;                                        01411
   IF getsub(topstd)=topstd THEN                                   01412
      abortsubsystem($"No substructure to this file!");
                                                                   01413
% Initialize the query stack areas. %                            01414
   qstrinit( entstd, topstd );                                     01415
   curindex ← entind;                                              01416
   confre ← 0;                                                     01417
   newstk ← TRUE;                                                  01418
IF shortcut THEN % a node has been specified. %                    01419
   BEGIN                                                           01420
   typead ← $name;                                                 01421
   RETURN( helpshow( &resultptr, parsing, $typead,
   &intparm));                                                     01422
   END;                                                            01423
% Print out the first node and menu and also the entry
message.  Initialize moremen: FALSE if no more to be
shown, TRUE otherwise. %                                           01424
   CASE qdisp(entstd) OF                                           01425
      = 1: % Everything OK; no more to be shown. %                 01426
         BEGIN                                                     01427
         moremen ← FALSE;                                          01428
         RETURN( &resultptr );                                     01429
         END;                                                      01430
      = -1: % More items to be shown; tell the user. %
                                                                   01431
         BEGIN                                                     01432
         moremen ← TRUE;                                           01433
```

```
                    RETURN( &resultptr );                                   01434
                    END;                                                    01435
                ENDCASE % Terrible error. %                                 01436
                    BEGIN                                                   01437
                    moremen ← FALSE;                                        01438
                    abortsubsystem($"Data base portrayal trouble.
                    Call ARC!");                                            01439
                    END;                                                    01440
        END;                                                                01441
    ENDCASE % Termination of HELP command; release storage if
    necessary. %                                                            01442
        BEGIN                                                               01443
        IF qagain THEN RETURN(&resultptr);                                  01444
        % redirect control-Q interrupt %                                    01445
          chntab[24] ← $gotohelp .V 186;                                    01446
        % Reset help flag %                                                 01447
          inhelp ← FALSE;                                                   01448
        IF &qsw THEN closeseq(&qsw := 0);                                   01449
        IF &qnewstmt THEN freestring(&qnewstmt:=0, $dspblk);                01450
        IF &qstorblk THEN freestring( &qstorblk:=0, $dspblk );              01451
        IF &hlpcmdstk THEN freestring( &hlpcmdstk:=0, $dspblk );
                                                                            01452
        IF &qda THEN                                                        01453
            BEGIN                                                           01454
            clearda(&qda);                                                  01615
            delda(&qda);                                                    01455
            IF nlmode = fulldisplay THEN                                    01456
                BEGIN                                                       01457
                % set bit in da's so that they are not ignored. %
                                                                            01458
                  end ← (&da ← $dpyarea) + dacnt*dal;                       01459
                  DO da.daauxiliary ← FALSE                                 01460
                  UNTIL (&da ← &da + dal) >= end;                           01461
                % Restore auxiliary bit for calculator display area.
                %                                                           01462
                  cda.daauxiliary ← calcaux ;                               01463
                dpset(dspallf, endfil, endfil, endfil);                     01464
                recred();                                                   01465
                END;                                                        01466
            END;                                                            01467
        dismes(0);                                                          01468
        END;                                                                01469
    RETURN(&resultptr);                                                     01470
    END.
                                                                            01471
(helpshow) PROC (resultptr, parsemode, type, param);                        01116
    LOCAL stid;                                                             01117
    REF resultptr, type, param;                                             01118
    CASE parsemode OF                                                       01119
        = parsing:                                                          01120
            BEGIN                                                           01121
            CASE type OF                                                    01122
                = $name: % param contains  a pointer to a string with
                either a number (menu) or a word to be used in a name
                search %                                                    01123
                    BEGIN                                                   01124
```

```
IF NOT qsearch(param.stpsid % string address %,
curindex: stid, curindex) THEN                                  01125
   BEGIN                                                        01126
   dismes(2,$"Item not found");                                 01127
   RETURN( &resultptr );                                        01128
   END                                                          01129
ELSE                                                            01130
   BEGIN                                                        01131
   IF stid # conrng[curindex*rnglnt] THEN                       01132
      BEGIN                                                     01133
      newstk ← TRUE;                                            01134
      IF NOT conrng[confre*rnglnt] THEN                         01135
      %we found a free ring element - bump index and
      confre %                                                  01136
         BEGIN                                                  01137
         curindex ← confre;                                     01138
         IF confre = rngmax THEN confre ← 0                     01139
         ELSE BUMP confre;                                      01140
         END                                                    01141
      ELSE % no empty ring elements so reuse the
      oldest %                                                  01142
         BEGIN                                                  01143
         curindex ← confre;                                     01144
         % go free up the stack we are going to use
         now %                                                  01145
         BUMP confre;                                           01146
         END;                                                   01147
      &curstk ← &conrng + curindex*rnglnt;                      01148
      curstk ← stid;                                            01149
      END                                                       01150
   ELSE newstk ← FALSE;                                         01151
   qda.dacsp ← stid; % so that includes will have the
   proper file for link parses %                                01152
   qda.daccnt ← 1;                                              01153
   END;                                                         01154
CASE qdisp(stid) OF                                             01155
   = 1: % Everything OK; no more to be shown. %                 01156
      BEGIN                                                     01157
      IF hseqer THEN abortsubsystem($"Ran out of
      space making menus.");                                    01158
      moremen ← FALSE;                                          01159
      dismes(0);                                                01160
      RETURN( &resultptr );                                     01161
      END;                                                      01162
   = -1: % Things OK for now, but must get more
   interaction from user. %                                     01163
      BEGIN                                                     01164
      IF hseqer THEN abortsubsystem($"Ran out of
      space making menus.");                                    01165
      moremen ← TRUE;                                           01166
      RETURN( &resultptr );                                     01167
      END;                                                      01168
   ENDCASE % Terrible error. %                                  01169
      BEGIN                                                     01170
      moremen ← FALSE;                                          01171
      abortsubsystem($"Data base portrayal trouble.
```

```
            Call ARC!");                                        01172
            END;                                                01173
      END;                                                      01174
= $rest: % show the rest of the menu that would not fit
%                                                               01175
      BEGIN                                                     01176
      IF NOT &qsw THEN                                          01177
         BEGIN                                                  01178
         % Display error message %                              01179
            dismes(2, $"Last menu was complete: No more");
                                                                01180
         END                                                    01181
      ELSE                                                      01182
         BEGIN % Call qdisp in continue mode. %                 01183
         CASE qdisp(0) OF                                       01184
            = 1: % Everything OK; no more to be shown. %
                                                                01185
               BEGIN                                            01186
               IF hseqer THEN abortsubsystem($"Ran out of
               space making menus.");                           01187
               moremen ← FALSE;                                 01188
               dismes(0);                                       01189
               RETURN(&resultptr);                              01190
               END;                                             01191
            = -1: % Things OK for now, but must get more
            interaction from user. %                            01192
               BEGIN                                            01193
               IF hseqer THEN abortsubsystem($"Ran out of
               space making menus.");                           01194
               moremen ← TRUE;                                  01195
               RETURN(&resultptr);                              01196
               END;                                             01197
            ENDCASE % Terrible error. %                         01198
               BEGIN                                            01199
               abortsubsystem($"Data base portrayal
               trouble.  Call ARC!");                           01200
               moremen ← FALSE;                                 01201
               END;                                             01202
         END;                                                   01203
      END;                                                      01204
= $back: % param contains the stid of the back node to
be displayed. %                                                 01205
      BEGIN                                                     01206
      curindex ← param;                                         01207
      stid ← param[1];                                          01208
      newstk ← FALSE;                                           01209
      qda.dacsp ← stid;                                         01210
      qda.daccnt ← 1;                                           01211
      CASE qdisp(stid) OF                                       01212
         = 1: % Everything OK; no more to be shown. %           01213
            BEGIN                                               01214
            IF hseqer THEN abortsubsystem($"Ran out of
            space making menus.");                              01215
            moremen ← FALSE;                                    01216
            dismes(0);                                          01217
            RETURN( &resultptr );                               01218
```

```
                    END;                                                01219
                  = -1: % Things OK for now, but must get more
                  interaction from user. %                              01220
                    BEGIN                                               01221
                    IF hseger THEN abortsubsystem($"Ran out of
                    space making menus.");                              01222
                    moremen ← TRUE;                                     01223
                    RETURN( &resultptr );                               01224
                    END;                                                01225
                  ENDCASE % Terrible error. %                           01226
                    BEGIN                                               01227
                    moremen ← FALSE;                                    01228
                    abortsubsystem($"Data base portrayal trouble.
                    Call ARC!");                                        01229
                    END;                                                01230
                END;                                                    01231
              = $menu: % param contains the stid of a menu item%        01232
                typeas($"notyet");                                      01233
              ENDCASE err($"Help system error:
              Invalid param passed to helpshow");                       01234
          END;                                                          01235
       ENDCASE;                                                         01236
    RETURN(&resultptr);                                                 01237
    END.
                                                                        01238
(checkmore) PROC(result, parsemode);                                     0934
    CASE parsemode OF                                                    0935
       = parsing :                                                       0936
          BEGIN                                                          0937
          IF moremen THEN RETURN(result)                                 0938
          ELSE RETURN(0);                                                0939
          END;                                                           0940
       ENDCASE;                                                          0941
    RETURN(result);                                                      0942
    END.
                                                                         0943
(rstmor) PROC(result, parsemode);                                        0944
    % Reinitialize moremen to FALSE; close sequences. %                 0945
    CASE parsemode OF                                                    0946
       = parsing :                                                       0947
          BEGIN                                                          0948
          moremen ← FALSE;                                               0949
          IF &qsw THEN                                                   0950
             BEGIN                                                       0951
             moreterm();  %  Must be at a lower level because of
             symbol conflicts. Closes sequences and resets viewspecs.
             %                                                          01299
             END;                                                        0958
          END;                                                           0959
       ENDCASE;                                                          0960
    RETURN(result);                                                      0961
    END.
                                                                         0962
(xhlpring) PROCEDURE( % provides feedback for stepping through the
rings for BACK command in help %                                        01551
    % FORMAL ARGUMENTS %                                               01552
```

```
    resultptr,     % ptr to result record %                          01553
    parsemode,     % parsing mode %                                  01554
    nwindex);                                                       01555
  LOCAL conad, stid, stdb, len;                                     01556
  LOCAL TEXT POINTER tp1, tp2;                                      01557
  LOCAL STRING temp[20];                                            01558
  REF resultptr, nwindex, conad;                                    01559
  % ------------- %                                                01560
  CASE parsemode OF                                                 01561
    = parsing:                                                      01562
      BEGIN                                                         01563
      % advance through the ring %                                  01564
      % If we are at the beginning, say we can go no further. %
                                                                    01565
        IF nwindex = entind THEN                                    01566
          BEGIN                                                     01567
          !sti(18M,CA);                                             01568
          resultptr ← curindex;                                     01569
          resultptr[1] ←                                            01570
            IF curindex = entind THEN entcon                        01571
            ELSE [&conrng + curindex*rng1nt];                       01572
          dismes( 2, $"No others have been shown");                 01573
          RETURN(&resultptr);                                       01574
          END;                                                      01575
        &conad ←                                                    01576
          IF nwindex = entind THEN $entcon                          01577
          ELSE &conrng + nwindex*rng1nt;                            01578
        IF nwindex # entind THEN                                    01579
          BEGIN                                                     01580
          nwindex ← conbck(nwindex);                                01581
          &conad ←                                                  01582
            IF nwindex = entind THEN $entcon                        01583
            ELSE &conrng + nwindex*rng1nt;                          01584
          END;                                                      01585
        stid ← conad;                                               01586
        %display the current statement%                             01587
          % display first 20 chars of stmt in name area %           01588
            % get statement length %                                01589
              lodsdb(getsdb(stid):stdb);                            01590
              len ← [stdb].schars + 1; % number of chars %          01591
            % construct text ptrs to each end of stmt %            01592
              tp1 ← stid;                                           01593
              tp1[1] ← 1;                                           01594
              IF NOT FIND SF(tp1) [EOL ↑tp2 ←tp2 /  "##" ↑tp2
              ←2tp2 ] THEN                                          01595
                BEGIN                                               01596
                tp2 ← stid;                                         01597
                tp2[1] ← len;                                       01598
                END;                                                01599
              % Truncate %                                          01600
                tp2[1] ← MIN(20, tp2[1]);                           01601
            % assign something to the string "temp" %              01602
              IF tp2[1] > 1 THEN *temp* ← tp1 tp2 ELSE *temp* ←
              "<NULL>";                                             01603
          dn($temp); %display string%                               01604
        % save the current state in the result record %            01605
```

```
            resultptr ← nwindex;                                    01606
            resultptr[1] ← stid;                                    01607
         END;                                                       01608
      = backup,                                                     01609
      = cleanup:                                                    01610
         dn( $""); % clear the name area %                          01611
      ENDCASE;                                                      01612
   RETURN( &resultptr );                                            01613
   END.
                                                                    01614
(qbkint) PROCEDURE (resultptr, parsemode);                          01287
   REF resultptr:                                                   01288
   % ------------- %                                               01289
   CASE parsemode OF                                                01290
      = parsing:                                                    01291
         BEGIN                                                      01292
         resultptr ← curindex;                                      01293
         END;                                                       01294
      ENDCASE;                                                      01295
   RETURN(&resultptr);                                              01296
   END.
                                                                    01297
FINISH of pshelp                                                     0253
```