```
< NLS, CPROGRAMS.NLS;17, >, 18-SEP-74 16:21 HGL ;;;;% invoke the TENLDR
saved in our address space to load a rel file
  the file is loaded into the user program buffer, and the tenldr
expects three arguments set up %
FILE cprograms % LIO <rel-nls>CPROGRAMS %% (110,)
(rel-nls,CPROGRAMS.rel,) %

DECLARE symloc=116B, nlsloader=117B, ddtsptr=770001B;
REGISTER r1 = 1, r2 = 2;  REF tda, symloc, nlsloader, ddtsptr;

%compile%
   (cpconan)   % content analyzer pattern compile routine %
      PROCEDURE(
      % FORMAL ARGUMENTS %
         tptr,          % ptr to content pattern string %
         da);  % ptr to display area %
      % RETURNS %
         % 1) address of the program in the programs buffer %
      REF tptr, da;
      LOCAL TEXT POINTER tptr2;
      LOCAL STRING patternstr[100];
      % ---------------- %
      *patternstr* ← tptr SE(tptr), " ;:";
      FIND SF(*patternstr*) ↑tptr2;
      RETURN (gpconan( $tptr2, &da ));
      END.

   (cpcmpfl)   %Core NLS Compile File command %
      PROCEDURE
         (type,
            % FALSE > cmpnam is adr of parsed link data structure %
               % also file name associated with link must have
               extenstion name = *savext* %
            % TRUE > cmpnam is adr of compiler name string %
         cmpnam,
         ofilnam,       % string containing the name of the output file
         %
         da);  % display area descriptor %

      LOCAL jfn, errors, cjfn;
      LOCAL TEXT POINTER tp;
      LOCAL STRING locstr[40];
      REF cmpnam, ofilnam, da;

      % open the output file %
         IF NOT jfn ← lgetjfn (0, &ofilnam, $relext, gtjoosf, $lit)
         THEN
            SIGNAL (ofilerr, $lit);
         IF NOT sysopen (jfn, write, bintyp, $lit) THEN
            BEGIN
            reljfn (jfn);
            SIGNAL (ofilerr, $lit);
            END;
      % get jfn for compiler if passed parsed link data structure %
         cjfn ← 0;
         IF NOT type THEN
```

```
        IF NOT (cjfn ← upgjfn( &cmpnam ) ) THEN
           SIGNAL( prcerr, $lit)
        ELSE
           BEGIN
           jfntostr( cjfn, $locstr, 000100B6);
           IF *locstr* # *savext* THEN
              SIGNAL( prcerr, $"Illegal Compiler Name")
           END;
   tp[0] ← da.dacsp; tp[1] ← 1;
   errors ←
      processor( IF type THEN &cmpnam ELSE 0,
         $tp, &da, cmptyp, jfn, cjfn);
   % close output file (but keep jfn) %
      IF NOT sysclose (jfn .V hBll, $lit) THEN
         BEGIN
         reljfn (jfn);
         dismes (2, $lit);
         GOTO STATE;
         END;
   % delete old versions, if no errors and no rubout %
      IF errors = 0 AND NOT inptrf THEN delovsrns (jfn, nvtk - 1);
      reljfn (jfn);
   RETURN (errors);
   END.

(cpcmproc)    PROCEDURE   % does Compile Procedure %
   (stid, % stid where compilation is to start %
   da):   % display area descriptor %

   % Compile a procedure.  If the compilation is successful and
   there is room in the buffer for the code and room in the user
   program stack, add the program to the stack of user programs, and
   do a procedure replace.  Mark DDT's symbol table. %

   LOCAL size, errs;
   LOCAL TEXT POINTER tp, tpb, tpe;
   LOCAL STRING tempstr[100], locstr[100];
   REF cm1tmp, da;

   IF NOT (FIND SF(stid) ↑tp [']() $NP ↑tpb $LD ↑tpe) THEN
      SIGNAL (upgerr, $"couldn't find procedure name");
   *lit* ← tpb tpe;
   *locstr* ← + tpb tpe;
   % setup to use special sequence %
      cm5tmp ← da.davspec;
      cm6tmp ← da.davspc2;
      cm3tmp ← da.dacacode;
      cm1tmp ← da.dausqcod;
      da.davspec.vsusqf ← da.davspec.vsbrof ← TRUE;
      da.dausqcod ← $cpseqg;
      da.dacacode ← 0;
      &cm1tmp ← 0;
      ON SIGNAL ELSE
         BEGIN
         da.davspec ← cm5tmp;
         da.davspc2 ← cm6tmp;
```

```
        da.dacacode ← cm3tmp;
        da.dausqcod ← cm4tmp;
        IF &cm1tmp THEN closeseq( &cm1tmp := 0 );
        END;
    marksym ($locstr);
    /upgffbuf/ ← upgbend - upgffbuf;   % space left in buffer %
    errs ← <SEQFIL, processor>
       (IF exp THEN $x110nam ELSE $110nam, $tp, &da, upgtyp,
       upgffbuf, 0  :size);
    % cleanup from above %
       da.davspec ← cm5tmp;
       da.davspc2 ← cm6tmp;
       da.dacacode ← cm3tmp;
       da.dausqcod ← cm4tmp;
       IF &cm1tmp THEN closeseq( &cm1tmp := 0 );
       ON SIGNAL ELSE;
    IF errs > 0 THEN
       BEGIN
       popsym();
       RETURN (errs)
       END
    ELSE IF upgskix >= $upgsksz THEN
       BEGIN
       popsym();
       SIGNAL (upgerr, $"no more room in user program stack")
       END
    ELSE
       % compilation was successful and there was room in the buffer
       and there is room in stack %
       BEGIN
       upgstk /upgskix ← upgskix + 1/ ← upgffbuf;
       upgffbuf ← upgffbuf + size;
       *upgnms* ← *upgnms*, SP, *locstr*;
       % do procedure replace %
          IF ddtlookup( $locstr, FALSE ) THEN
             BEGIN
             IF rplproc( $locstr, $locstr) THEN
                BEGIN
                *tempstr* ← "Procedure ", *locstr*, " Replaced";
                dismes(2, $tempstr);
                END;
             END;
       END;
    RETURN (0);
    END.

% sequence generator program for compile procedure %
    (cpseqg) PROCEDURE (sw, which); % Compile Procedure Seq Generator
    %
       LOCAL vpc;
       LOCAL STRING locstr/200/;
       REF cm1tmp, sw;
       CASE which OF
          =sqopn: % called at open %
             BEGIN
             vpc ← cm5tmp;  vpc.vsbrof ← TRUE;
```

```
            cm2mp ← 0;
            &cm1tmp ←
               openseq( sw.swcstid, sw.swlbstid, vpc, cm6tmp,
               cm4tmp, cm3tmp);
            END;
         =sqgnxt: % called for next in seq -- fall through %
            IF FALSE = (cm2mp := TRUE) THEN
               BEGIN
               *locstr* ← "PROGRAM ", *lit*;
               send( &sw, $locstr);
               REPEAT CASE( sqgnxt );
               END
            ELSE
               LOOP
                  BEGIN
                  IF (sw.swstid ← seqgen(&cm1tmp) ) = endfil THEN
                     send( &sw, $"FINISH");
                  cpysw( &cm1tmp, &sw);
                  sport( &sw );
                  END;
         =sqcls: % called at close %
            BEGIN
            closeseq( &cm1tmp := 0 );
            END;
         ENDCASE err ($"Illegal call to a seq gen program"); %
         called for any other purpose -- error %
      RETURN;
      END.

(dinstupg) PROCEDURE   %  Deinstitute a program %
   (pgmaddr);   % address of of the user program %
   % "deinstitute" the specified user program --
      find any references to it and delete them, but don't give back
      its space to the buffer -- thus it can "reinstituted" %
   LOCAL da;
   REF da:
   FOR &da ← $dpyare UP da1 UNTIL >= $dpyare + dacnt*da1 DO
      IF da.daexis THEN
         BEGIN  % fields pointing to program to be popped are reset %
         IF da.dacacode = pgmaddr THEN da.dacacode ← 0;
         IF da.dausqcod = pgmaddr THEN da.dausqcod ← 0;
         IF da.daukeycod = pgmaddr THEN da.daukeycod ← 0;
         END;
   RETURN:
   END.

(gpbsz)  PROCEDURE  % Goto User program Buffer Size Set %
   (size);  % Number of pages to be in new buffer %

   % RETURNS %
      % returns false for invalid request or address of first new page
      allocated %
   LOCAL stadr;

   IF rfpmax - size < 20 OR (512*size) < (upgffbuf-upgbuf+1)
      THEN RETURN (FALSE);
```

```
    closeall();
    upgbsz ← size;
    rfpmin ← size + 1;  % Index to first file core page %
    stadr ← upgbend + 1;
    upgbend ← upgbuf + 512*size - 1;
    openall();
    RETURN (stadr);
    END.

(gpgmrst)  PROCEDURE;  % Goto User program reset %
    % reset the stack and all display area descriptor fields having to
    do with user programs %
    %reset user programs buffer size to default value%
    ON SIGNAL
        = upgerr: RETURN;
        ELSE;
    LOOP gppop();
    RETURN:
    END.

(gpconan)  PROCEDURE   % does Goto Program Content analyzer %
    (tpb.  % text pointer to string containing pattern %
    da):       % display area %

    % Compile a content analyzer pattern.  If the compilation is
    successful and there is room in the buffer for the code and room in
    the user program stack, add the pattern to the stack of user
    programs -- errors cause a upgerr SIGNAL to be generated.  DDT's
    symbol table is marked. %

    LOCAL size, errs, location:
    LOCAL TEXT POINTER tpe:
    REF tpb, da;

    marksym (0);  % mark the symbol table %
    /upgffbuf/ ← upgbend - upgffbuf;    % space left in buffer %
    errs ← <SEQFIL, processor>
        (IF exp THEN $x11onam ELSE $11onam, &tpb, &da, upgtyp, upgffbuf,
        0  :size);
    IF errs # 0 THEN
        BEGIN
        popsym();
        SIGNAL (upgerr, errs)
        END
    ELSE IF upgskix >= $upgsksz THEN
        BEGIN
        popsym();
        SIGNAL (upgerr, $"no more room in user program stack")
        END
    ELSE
        % compilation was successful and there was room in the buffer and
        there is room in stack %
        BEGIN
        location ← upgstk /upgskix ← upgskix + 1/ ← upgffbuf;
        upgffbuf ← upgffbuf + size;
        BUMP upgcacnt;
```

```
      FIND > tpb SNP ↑tpb $5 PT ↑tpe;
      *upgnms* ← *upgnms*. SP. "UP", STRING(upgcacnt), '1, tpb tpe;
      END:
   RETURN (location);
   END.

(gpexpgm)  PROCEDURE   % does Goto Program Execute a program %
   (str);   % string containing name or number of the user program %
   % control is transferred to specified program by means of a CALLO%
   LOCAL upgaddr;
   REF str;
   upgaddr ← upgcnv (&str);
   IF NOT upgaddr > 0 THEN
      SIGNAL (upgerr, $"no address for program");
   /upgaddr/();
   RETURN:
   END.

(gpget)    PROCEDURE   % does Goto Programs Get Rel  File command %
   (adstr.   % address of parsed link data structure %
   shomes);  % If TRUE, do dismes's; FALSE, only error mssages form
   loader get out. %
   % invoke the TENLDR saved in our address space to load a rel file
     the file is loaded into the user program buffer, and the tenldr
   expects three arguments set up %
   LOCAL saveprogaddr, savsymloc, jfn, errs, lc, loader, entry, ddtloc,
   oneflag, extflag, pgmfield, try, stname, i, ptype, dtstr[40];
   LOCAL TEXT POINTER tps,tpe, tp1, tp2:
   LOCAL STRING errstr[200]:
   LOCAL STRING recurse[200];
   LOCAL STRING filename[39];
   LOCAL STRING extname[39]:
   LOCAL STRING tempstr[50];
   LOCAL STRING extension[10];
   LOCAL STRING str[200];
   REF adstr, loader;
   % initialize %
      oneflag ← TRUE;
   % get the file name specified by the parsed link data structure %
      CASE lnpfls( 0, &adstr, $str) OF
         = lhostn:  NULL;
         ENDCASE err($"Remote File Manipulations Not Implemented Yet");
   % edit the name string to get just the file name %
      extflag ← FALSE; % set to TRUE by the FIND stmt if extension is
      given %
      FIND SF(*str*) ↑tps ↑tpe;
      IF FIND tpe >  '< ['> / '<↑F> / '<ESC>] ↑tps THEN NULL
      ELSE IF NOT FIND tpe > LD THEN
         err($"Illegal Rel File Name");
      IF FIND tps > ['.] ↑tpe ←tpe THEN
         BEGIN
         IF FIND LD THEN extflag ← TRUE:
         END
      ELSE
         IF FIND tps > ['<↑F> / '<ESC>] ↑tpe THEN
            BEGIN
```

```
      IF FIND LD THEN extflag ← TRUE;
      END
   ELSE
      IF NOT FIND tps > $LD (EOL / ENDCHR) ↑tpe THEN
         err($"Illegal Rel File Name");
  *filename* ← tps tpe;
% save away current state values %
  savsymloc ← symloc;
% get the jfn for the rel file %
  % all of the following rather obtuse code attempts to figure out
  what file is to be loaded and what type it is.  It is not
  necessary for the user to specify the extension name, as this
  grungy code manages to find that out.  If no extension is given,
  then extensions are attempted in the following order: REL, CA,
  SK, SG, SUBSYS, CML, PROC-REP.  If this sequence cannot be
  satisfied in the default directory for links (for bugged links if
  no directory is specified) or in the connected directory (for
  typed in links if no directory is specified) then the PROGRAMS
  directory is used and the same sequence is repeated %
  *extension* ← "REL"; try ← 0;
  CASE jfn ← lgetjfn(0,$str,$extension,1B11,$errstr) OF
     =FALSE: % getjfn failed, try some recovery strategy %
        BEGIN
        WHILE NOT extflag DO
           BEGIN % try to figure out what extension user wants %
           CASE try ← try + 1 OF
              =1: *extension* ← "CA";
              =2: *extension* ← "SK";
              =3: *extension* ← "SG";
              =4: *extension* ← "SUBSYS";
              =5: *extension* ← "CML";
              =6: *extension* ← "PROC-REP";
              ENDCASE EXIT LOOP;
           REPEAT CASE;
           END;
        % maybe try default user program directory %
           IF (oneflag := FALSE) AND
              ((*str*[1] # '<) OR (adstr[ue+1] = adstr[us+1])) THEN
                 BEGIN
                 *str*  ← "<PROGRAMS>", tps SE(*str*);
                 *extension* ← "REL"; try ← 0;
                 REPEAT CASE;
                 END;
        err($errstr);
        END;
  ENDCASE;
% get rid of any ↑F or altmodes in filename %
  jfntostr( jfn, $filename, 001000B6);
  jfntostr( jfn, $extname, 000100B6);
% find out what type of program for future instituting %
  ptype ← 0;
  IF *extname* = "CA" THEN ptype ← 1
  ELSE
     IF *extname* = "SK" THEN ptype ← 2
     ELSE
        IF *extname* = "SG" THEN ptype ← 3
```

```
         ELSE
            IF *extname* = "PROC-REP" THEN ptype ← 4
            ELSE
               IF *extname* = "CML" THEN ptype ← 5
               ELSE
                  IF *extname* = "SUBSYS" THEN ptype ← 6
                  ELSE NULL;
% open the file %
   IF NOT SKIP !openf (jfn, 440000285) THEN
      BEGIN
      IF NOT SKIP !rljfn( jfn ) THEN NULL;
      err( $"Open File Failed" );
      END;
%adjust the user program buffer size if necessary to fit the program
being loaded%
   gpadjbsz(jfn);
% invoke the tenldr %
   IF shomes THEN dismes(1, $"Loading User Program");
   &loader ← nlsloader;
   errs ← loader(jfn, upgffbuf+1, upgbend :lc);
   IF shomes THEN dismes(0);
% the tenldr returns an error count and the next address in the
buffer %
   IF NOT errs THEN % good return %
      BEGIN
      % mark the block in the user program buffer %
         IF upgskix >= $upgsksz THEN
            SIGNAL (upgerr, $"no more room in user program stack")
         ELSE
            % load was successful and there was room in the buffer
            and there is room in stack %
            BEGIN
            upgstk [upgskix ← upgskix + 1] ← upgffbuf;
            % set string into user program buffer %
               *upgnms* ← *upgnms*, SP, *filename*;
            END;
      % update ddt's alt I-1 to include program just loaded %
         ddtloc ← ddtsptr; % get ptr to alt i -1 %
         [ddtloc] ← symloc: % stuff away new symloc %
      % mark the block in the symbol table  %
         ddmrk[ddmrkx] ← savsymloc.RH - 1;
         IF (ddmrkx ← ddmrkx + 1 ) > ddmrkm THEN
            BEGIN
            ddmrkx ← ddmrkx - 1;
            err($"Mark stack overflow");
            END;
      % lookup file name and set entry instruction %
         IF stname ← ddtlookup( $filename, FALSE : entry ) THEN
            [upgffbuf] ← 254B9 + entry %set up entry inst %
         ELSE
            CASE ptype OF
               =0, =1, =2, =3:      % REL, CA, SK, SG %
                  IF shomes THEN dismes(2,$"WARNING -- no entry to
                  program ");
               =4:                  % PROC-REP %
                  BEGIN
```

```
            *tempstr* ← "WARNING -- No Procedure Named ",
            *filename*;
            IF shomes THEN dismes(2,$tempstr);
            END;
         =5:                    % CML %
            BEGIN
            *tempstr* ← "WARNING -- No Subsystem Named ",
            *filename*;
            IF shomes THEN dismes(2,$tempstr);
            END;
         =6:                    % SUBSYS %
            IF shomes THEN dismes(2,$"Don't Execute via RUN
            PROGRAM Command
            Use GOTO SUBSYSTEM Command");
         ENDCASE;
% set new limit on buffer space %
   saveprogaddr ← upgffbuf := lc;
% institute program if so indicated by extension %
   CASE ptype OF
      = 1: % content analyzer %
         BEGIN
         IF shomes THEN dismes(2,$"Instituting User Program as
         a Content Analyzer");
         [lda()].dacacode ← saveprogaddr;
         END;
      = 2: % sort key %
         BEGIN
         IF shomes THEN dismes(2,$"Instituting User Program as
         a Sort Key Program");
         [lda()].daukeycod ← saveprogaddr;
         END;
      = 3: % sequence generator %
         BEGIN
         IF shomes THEN dismes(2,$"Instituting User Program as
         a Sequence Generator");
         [lda()].dausqcod ← saveprogaddr;
         END;
      = 4: % procedure replace %
         IF stname THEN
            BEGIN
            IF rplproc( $filename, $filename) THEN
               *tempstr* ← "Procedure ", *filename*, "
               Replaced"
            ELSE
               *tempstr* ← "Old Procedure ", *filename*, "
               Does Not Exist";
            IF shomes THEN dismes(2, $tempstr);
            END;
      = 5:    % cml %
         IF stname THEN
            BEGIN
            upgstk[upgskix].LH ← entry;
            dfnsubsys( entry, gtctrlbits(entry), $nlssubs );
            dfnsubsys( entry, gtctrlbits(entry), $allsubs );
            *tempstr* ← "Subsystem "=@ +filename*, " Now
            Available (Attached)";
```

```
                IF shomes THEN dismes(2, $tempstr);
                END;
            = 6:    % subsys %
                BEGIN
                *tempstr* ← *filename*, ".CML";
                FOR i ← 0 UP UNTIL = 40 DO dtstr[i] ← adstr[i];
                FIND SF(*tempstr*) ↑tp1 SE(*tempstr*) ↑tp2;
                dtstr[fs] ← tp1: dtstr[fs+1] ← tp1[1];
                dtstr[fe] ← tp2: dtstr[fe+1] ← tp2[1];
                ON SIGNAL ELSE
                   BEGIN
                   ON SIGNAL ELSE;
                   err($"Can't load frontend for this SUBSYS");
                   END;
                gpget($dtstr, shomes);
                ON SIGNAL ELSE;
                %upgstk[upgskix-1].LH ← upgstk[upgskix].LH;%
                %mark the LH of the second from the top entry of the
                user program stack with -1 . This indicates that is a
                user .subsys program which has had a corresponding
                user .cml program automatically loaded on top of it.
                This flag is used by gppop so that the command Delete
                Last (program in buffer) will actually delete both
                the CML and Subsys portions of the user subsystem%
                upgstk[upgskix-1].LH ← -1;
                %BUMP DOWN upgskix;%
                %popsym(); This has been removed so that detach
                subsystem will work properly. This implies that when
                loading a subsystem both the CML and the SUBSYS
                symbols will be on the symbol stack. These two blocks
                in the symbol table should therefore have distinct
                names (distinct names on the file statements)%
                FIND SE(*upgnms*) $PT $NP ↑tp2;
                *upgnms* ← SF(*upgnms*) tp2;
                END;
            ENDCASE;
        END
    ELSE
        BEGIN
        symloc ← savsymloc; % reset symloc as it may be clobbered by
        an error in loading %
        err( $"Error in Loading");
        END;
RETURN;
END.

(gtctrlbits) PROCEDURE  % return ctrl bits for new subsystem %
    (dptptr);
    LOCAL instptr;
    REF dptptr, instptr;
    % check to make sure that we've been given a valid pointer %
        IF dptptr.dptvalid # $dptvldationcode THEN
            RETURN( FALSE );
    % set up to go through rule nlssubs %
        &instptr ← $nlssubs;
    %if there are no defined subsystems then return 7%
```

```
      IF instptr.opcode # $xecute THEN RETURN( 7 );
      &instptr ← instptr.addr;
   % loop through rule %
      WHILE &instptr DO
         IF instptr.opcode = $keyop THEN
            IF *[instptr.addr]* = *[dptptr.dptname]* THEN
               RETURN( instptr.ctrl )  % replacing existing subsys %
            ELSE IF *[instptr.addr]*[1] = *[dptptr.dptname]*[1] THEN
               % first char conflict %
                  IF instptr.ctrl.llcmd THEN RETURN( 3 )
                  ELSE &instptr ← instptr.alternative
         ELSE &instptr ← instptr.alternative;
   % not found so return all bits on %
      RETURN( 7 );
   END.

(gpadjbsz) PROCEDURE % adjusts user program buffer size %
   (jfn); %jfn of file which is going to be loaded%
   LOCAL
      flength, %byte size of rel file, used as estimate of core size%
      wordsleft,  %in user program buffer%
      wordsneeded, %to load this program (a reasonable guess)%
      wordstoadd,
      pagestoadd,
      remainder,
      newbsize;
   % Compute space left in user buffer %
      wordsleft ← upgbend - upgffbuf + 1;
   % Estimate core size of the rel file%
      !gtfdb(jfn,1B6+$fdbsiz,$flength);
      wordsneeded ← flength;
   %adjust buffer size so it will fit (hopefully)%
      IF wordsneeded < wordsleft THEN RETURN(TRUE);
      wordstoadd ← wordsneeded - wordsleft;
      pagestoadd ← (wordstoadd + 511) / 512;
      newbsize ← upgbsz + pagestoadd;
      gpbsz(newbsize);
      RETURN(TRUE);
   END.
(gpllo)    PROCEDURE   % does Goto Program L10 program compile %
   (stid, % stid where compilation is to start %
   da):   % display area descriptor %

   % Compile a L10 program.  If the compilation is successful and there
   is room in the buffer for the code and room in the user program
   stack, add the program to the stack of user programs, but do not
   make it the current CONAN program.  Mark DDT's symbol table. %

   LOCAL size, errs;
   LOCAL TEXT POINTER tp, tpb, tpe;
   LOCAL STRING str[50];
   REF da;

   IF NOT (FIND SF(stid) ↑tp [("PROGRAM"/"FILE")] $NP ↑tpb $LD ↑tpe)
   THEN
      SIGNAL (upgerr, $"couldn't find program name");
```

```
    *str* ← tpb tpe;
    astruc( $str );
    marksym ($str);
    /upgffbuf/ ← upgbend - upgffbuf;    % space left in buffer %
    errs ← <SEQFIL, processor>
       (IF exp THEN $x11Onam ELSE $11Onam, $tp, &da, upgtyp, upgffbuf, 0
        :size);
    IF errs > 0 THEN
       BEGIN
       popsym();
       RETURN (errs)
       END
    ELSE IF upgskix >= $upgsksz THEN
       BEGIN
       popsym();
       SIGNAL (upgerr, $"no more room in user program stack")
       END
    ELSE
       % compilation was successful and there was room in the buffer and
       there is room in stack %
       BEGIN
       upgstk /upgskix ← upgskix + 1/ ← upgffbuf;
       upgffbuf ← upgffbuf + size;
       *upgnms* ← *upgnms*, SP, *str*;
       END:
    RETURN (0);
    END.

(gppop)    PROCEDURE;  % does Goto Program Pop a program %
    % Delete top program on user program stack --
       pop DDT's symbol table
       find any references to it and delete them
       give back its space to the buffer
       fix the stack pointer and the string of program names %
    LOCAL oldend, i, value, bpa;
    LOCAL STRING pname/40/;
    LOCAL TEXT POINTER tp, tp1, tp2;
    IF upgskix <= 0 THEN SIGNAL (upgerr, $"stack is empty");
    FIND SE(*upgnms*) $NP ↑tp2 $PT ↑tp1 $NP ↑tp;
    % get name of last program (not needed) %
       % *pname* ← tp1 tp2;) %
    % detach the subsystem %
       ON SIGNAL ELSE GOTO gpgtl;
       IF (value ← upgstk/upgskix/.LH := 0) THEN
          BEGIN
          delsubsys(value, $nlssubs);
          delsubsys(value, $allsubs);
          END;
       (gpgtl): ON SIGNAL ELSE;
    popsym ();
    dinstupg (upgstk/upgskix/);
    IF upgstk/upgskix/ IN /upgbuf, upgbend) THEN
       % if program is in buffer, free the buffer space %
       oldend ← upgffbuf := upgstk/upgskix/;
    BUMP DOWN upgskix;
    % clear any breakpoints here %
```

```
    FOR i ← 0 UP UNTIL >= 10 DO
       IF bpa ← findbpa[i) THEN
          IF ([[bpa]].LH = 265B3 % JSP %) AND
             ([bpa].RH IN [upgffbuf, oldend]) THEN
                BEGIN
                [[bpa]] ← [bpa+1];
                [bpa] ← [bpa+1] ← 0;
                END;
 % do procedure backups for any procedures in this program %
    FOR i ← 0 UP 2 UNTIL >= 10 DO
       IF proctbl[i] THEN
          IF ([proctbl[i]].LH = 254B3) AND
             ([proctbl[i]].RH IN [upgffbuf, oldend]) THEN
                BEGIN
                [proctbl[i]] ← proctbl[i+1];
                proctbl[i] ← 0;
                END;
 IF upgstk[upgskix].LH  =18M THEN %was the program now on the top of
 the stack loaded as a subsys which automatically pulled in a cml
 over it? if so we should delete the subsys now because we just
 deleted the cml.%
    BEGIN
    upgstk[upgskix].LH ← 0;
    gppop();
    END
 ELSE *upgnms* ← SF(*upgnms*) tp;
 RETURN:
 END.

(gpstatus) PROCEDURE   % does Goto Program Status display %
    (str, da);   % address of string in which to put stuff %
    %make up a string containing the status of the user program stuff%
    REF str, da;
    *str* ←
       "Stack of compiled programs (first is #1):", EOL,
          "   ", *upgnms*, EOL, EOL;
    *str* ← *str*,
       "Content Analyzer  ", " program for display area:  ";
    upgsstr (da.dacacode, &str);
    *str* ← *str*,
       "Sequence Generator", " program for display area:  ";
    upgsstr (da.dausqcod, &str);
    *str* ← *str*,
       "Sort Key Extractor", " program for display area:  ";
    upgsstr (da.daukeycod, &str);
    *str* ←
       *str*, EOL, EOL,
       "Current buffer size: ", STRING(upgbsz), " pages = ", STRING
       (512*upgbsz), " words.  ", ECL,
       "Room left in buffer:  ",
          STRING (upgbend - upgffbuf + 1),  " words.", EOL;
    RETURN:
    END.

    (upgsstr) PROCEDURE (pgmaddr, str):
       LOCAL i;
```

```
    LOCAL TEXT POINTER tp1, tp2;
    LOCAL STRING lstr[50];
    REF str;
    IF pgmaddr = 0 THEN *lstr* ← "None"
    ELSE
       BEGIN
       *lstr* ← "None";
       FOR i ← upgskix DOWN UNTIL = 0 DO
          IF upgstk[i] = pgmaddr THEN
             BEGIN
             FIND SF(*upgnms*);
             FOR i ← i - 1 DOWN UNTIL = 0 DO FIND $NP $PT;
             FIND $NP ↑tp1 $PT ↑tp2;
             *lstr* ← tp1 tp2;
             EXIT LOOP 1;
             END;
       END;
    *str* ← *str*, *lstr*, EOL;
    RETURN;
    END.

(gptxst)        % does: SHOW TENEX SUBSYTEM STATUS %
   PROCEDURE
      ( fork,   % fork handle %
      astr      % address of string to receive status message %
      );
   LOCAL
      status,   % inferior fork's status %
      pc;       % inferior fork's pc %
   REF astr;

   % RETURNS %
      % returns FALSE for bad fork handle input %

   IF NOT fork THEN
      BEGIN
      *astr* ← "Fork Does Not Exist";
      RETURN( FALSE );
      END;
   !rfsts( fork );
   IF r1.LH = 777777B THEN
      BEGIN
      *astr* ← "Fork Does Not Exist";
      RETURN( FALSE );
      END;
   status ← r1.LH .A 377777B;
   pc ← r2.RH;
   IF r1 .A 4B11 THEN *astr* ← "Interrupted from ";
   CASE status OF
      = 0: *astr* ← *astr*, "Running";
      = 1: *astr* ← *astr*, "IO Wait";
      = 2: *astr* ← *astr*, "Halted";
      = 3:
         BEGIN
         *astr* ← *astr*, "Halted Because ";
         CASE r1.RH OF
```

```
            IN [0, 5], IN [12, 14], IN [21, 35], = 8:
               *astr* ← *astr*, "Channel ", STRING( r1.RH), "
               Interrupt";
            = 6:
               *astr* ← *astr*, "Overflow";
            = 7:
               *astr* ← *astr*, "Floating Overflow";
              9:
               *astr* ← *astr*, "Pushdown Overflow";
            = 10:
               *astr* ← *astr*, "End of File";
            = 11:
               *astr* ← *astr*, "IO Data Error";
            = 15:
               *astr* ← *astr*, "Illegal Instruction";
            = 16:
               *astr* ← *astr*, "Illegal Memory Read";
            = 17:
               *astr* ← *astr*, "Illegal Memory Write";
            = 18:
               *astr* ← *astr*, "Illegal Memory Execute";
            = 19:
               *astr* ← *astr*, "Fork Termination Interrupt";
            = 20:
               *astr* ← *astr*, "Disk Space Allocation Exceeded";
            ENDCASE;
         END;
      = 4: *astr* ← *astr*, "Fork Wait";
      = 5: *astr* ← *astr*, "Sleep";
      = 6: *astr* ← *astr*, "Breakpoint";
      = 100: *astr* ← *astr*, "Processing Suspended";
      ENDCASE;
   *astr* ← *astr*, " at ", STRING(pc, 8);
   RETURN( TRUE );
   END.

(gpkill)        % does: KILL TENEX SUBSYSTEM %
   PROCEDURE;
   % deactivate inferior termination psi %
      IF chntab[19] = $fkterm THEN
         BEGIN
         !dic( 400000B, 2B5);
         chntab[19] ← 0;
         END;
   % now get rid of the inferior %
      IF infork THEN !kfork( infork := 0 );
   % close and release any lingering jfns %
      IF pjfn THEN      % subsystem %
         BEGIN
         IF NOT SKIP !closf( pjfn ) THEN NULL;
         IF NOT SKIP !rljfn( pjfn := 0 ) THEN NULL;
         END;
      IF ojfn > 0 THEN          % output file %
         BEGIN
         IF NOT SKIP !closf( ojfn ) THEN NULL;
         IF NOT SKIP !rljfn( ojfn := 0 ) THEN NULL;
```

```
      END;
   IF ijfn > 0 THEN          % input file %
      BEGIN
      IF NOT SKIP !closf( ijfn ) THEN NULL;
      IF NOT SKIP !rljfn( ijfn := 0 ) THEN NULL;
      END;
% all done %
   RETURN;
END.

%  %
```

```
(gprunt)        % does: RUN TENEX SUBSYSTEM %
   PROCEDURE
      (adstr,   % parsed link data structure for subsystem to be run %
      rhosto,   % host number for output file %
      outfil,   % null string for tty output; else file name %
      mode,     % 3 - file: 1 - interactive; 2 - typeahead; 4 - none %
      rhostt,   % host number for input or typeahead file %
      tahd,     % file name / typeahead / termination character %
      wtmode    % TRUE for wait for completion %
      ):
   LOCAL
      intflg;   % flag indicating current status %
   LOCAL STRING
      tstring[40],      % temp string %
      locstr[500]:      % temp string %
   REF subnam, outfil, tahd, adstr;

   % can only run one inferior at a time %
      IF infork THEN
         BEGIN
         dismes( 2, $"Tenex Subsytem Already Running");
         RETURN;
         END;
   % do some intializing and state saving %
      !rfmod( 100B );
         tmode ← r2;
      !getnm();
         nlssbn ← r1;
      trmcod ← 0;
   % get a jfn for the subsystem first %
      IF NOT ( pjfn ← upgjfn(&adstr) ) THEN
         err($"Can't get jfn for this subsystem");
   % get jfn and open output file if asked for %
      IF NOT outfil.L THEN ojfn ← -1
      ELSE
         CASE rhosto OF
            = lhostn:
               BEGIN
               IF NOT (ojfn ← sgtjfn( gtjoof, &outfil, $lit)) THEN
                  err($"Can't get jfn for output file");
               IF NOT sysopen( ojfn, append, chrtyp, $lit) THEN
                  err($"Can't open output file");
               END;
            ENDCASE
               err($"Remote File Manipulations Not Implemented Yet");
   % get terminating character or %
   % open input file or %
   % create temporary input file if asked for %
      CASE mode OF
         = 3:  % input from file mode %
            CASE rhostt OF
               = lhostn:
                  BEGIN
                  IF NOT (ijfn ← sgtjfn( gtjoif, &tand, $lit)) THEN
                     err($"Can't get jfn for input fil#");
                  IF NOT sysopen( ijfn, read, chrtyp, $lit) THEN
```

```
                err($"Can't open input file");
              END;
           ENDCASE
              err($"Remote File Manipulations Not Implemented
              Yet");
     = 1:  % interactive mode %
        BEGIN
        ijfn ← -1;
        % check for valid termination character %
           CASE (intflg ← *tahd*[1]) OF
              IN [1,33]: NULL:     % ↑A through ↑Z %
              = 40: intflg ← 29;   % space %
              = 177: intflg ← 28;  % rubout %
              = ENDCHR: intflg ← 0;        % no termination char %
              ENDCASE
                 err($"Illegal Temination Character Specified");
        END;
     = 2, = 4:  % none or typeahead mode %
        BEGIN
        mode ← 2;  % look like typeahead from here on out %
        IF NOT tahd.L THEN ijfn ← 377777   % nil input %
        ELSE
           CASE rhostt OF
              = lhostn:
                 BEGIN
                 % get string name for temporary file %
                    jfntostr( pjfn, $tstring, 001000B6);
                    *locstr* ← '<, *userstr*, '>, *tstring*,
                    "-TYPEAHEAD.", *initsr*, ";T";
                 IF NOT (ijfn ← sgtjfn( gtjoof, $locstr, $lit))
                 THEN
                    err($"Can't get jfn for temporary input file");
                 IF NOT sysopen( ijfn, readwrite, chrtyp, $lit)
                 THEN
                    err($"Can't open temporary input file");
                 !sout( ijfn, chbmty+&tahd, -tahd.L);
                 IF NOT SKp !sfptr( ijfn, 0 ) THEN
                    err($"Can't set byte pointer for temporary
                    input file");
                 END;
              ENDCASE
                 err($"Remote File Manipulations Not Implemented
                 Yet");
        END;
     ENDCASE err($"Illegal Input Mode");
% create the inferior fork %
   IF NOT SKIP !cfork( 2B11 ) THEN
      err($"Can't create inferior fork");
   infork ← r1;
% enable all capabilities %
   !epcap( infork, -1, -1 );
% get the subsystem (and get rid of jfn ) %
   r1.LH ← infork;
   r1.RH ← pjfn;
   !get( r1 );
   IF NOT SKIP !r1jfn( pjfn ) THEN NULL;
```

```
% now start it; for interactive mode, shut down dnls first %
   CASE mode OF
      = 3, =2:       % file or typeahead mode %
         BEGIN
         % setup primary jfns %
            r2.LH ← ijfn;
            r2.RH ← ojfn;
            !spjfn( infork, r2);
         % set up to receive fork termination psi %
            IF NOT wtmode THEN
               BEGIN
               chntab[19] ← 1B6 + $fkterm;
               !aic( 400000B, 200000);
               END;
         % disable all terminal psi for inferior %
            !stiw( infork, 0);
         % start the fork %
            !sfrkv( infork, 0);
         % return (after waiting if appropriate %
            IF wtmode THEN
               BEGIN
               !wfork( infork );
               gpkill();
               !setnm( nlssbn );
               !sfmod( 100B, tmode := 0 );
               END;
            RETURN;
         END;
      = 1:           % interactive mode %
         BEGIN
         % shut down dnls %
            IF nlmode = fulldisplay THEN shutdis();
         % save terminal mode %
            !rfmod( 100B );
            tmode ← r2;
         % reset terminal modes %
            !sfmod( 100B, 20510175520B);
         % set up psi system %
            IF (trmcod ← intflg) THEN
               BEGIN  % termination char specified %
               % set up to receive fork termination psi %
                  IF NOT wtmode THEN
                     BEGIN
                     chntab[19] ← 1B6 + $fkterm;
                     !aic( 400000B, 200000);
                     END;
               % save terminal interrupt word %
                  !rtiw( 400000B );
                  tiw ← r2;
               % setup termination char as psi on channel 35 %
                  chntab[35] ← 1B6 + $trmpsi;
                  r1.LH ← trmcod;
                  r1.RH ← 35;
                  !ati( r1 );
                  !aic( 400000B, 1);
               % disallow all psi chars (for us) except terminating
```

```
                character %
                   r1 ← 35 - trmcod;
                   r2 ← 1;
                   !LSH r2,0(r1);
                   !stiw( 400000B, r2);
                % make sure we get the psi and not the inferior %
                   !sircm( infork, 1);
                END
             ELSE
                BEGIN  % no termination char specified %
                % save terminal interrupt word %
                   !rtiw( 400000B );
                   tiw ← r2;
                % disallow all psi chars (for us) except ↑C %
                   !stiw( 400000B, 0);
                END;
          % now start the fork %
             !sfrkv( infork, 0);
          END;

       ENDCASE;
% now wait for terminating char psi or until the fork finishes %
   % (get here only for interactive mode) %
   IF wtmode OR NOT trmcod THEN
      BEGIN          % no termination char specified %
      % wait for inferior to terminate %
         !wfork( infork );
      % resume nls %
         gprnls( tmode := 0 );
      % get back our psi tiw %
         !stiw( 400000B, tiw := 0 );
      % give user termination status %
         gptxst( infork, $locstr);
         dismes(2, $locstr );
      % now cleanup %
         gpkill();
      % and we are done! %
         RETURN;
      END
   ELSE      % termination char specified %
      !wait();       % this does not return!!! %

% we get here only when we get the terminating char psi %
   (trmpsi):
   % freeze the inferior to avoid some problems %
      !ffork( infork );
   % deactivate and deassign this psi %
      !dti( trmcod := 0 );
      !dic( 400000B, 1);
      chntab[35] ← 0;
   % setup primary jfns %
      !gpjfn( infork );
      IF r2.LH = 777777B THEN r2.LH ← 377777B;
      r2.RH ← IF ojfn > 0 THEN ojfn ELSE 777777B;;
      !spjfn( infork, r2);
   % disable all terminal psi for the inferior %
```

```
      !stiw( infork, 0);
   % get back our proper tiw %
      !stiw( 400000B, tiw := 0);
   % now get back into NLS %
      gprnls( tmode );
   % setup to debrk to sysrtn %
      IF NOT wtmode THEN [levtab] ← $sysrtn;
   % now resume the inferior %
      !rfork( infork );
   % now debrk %
      !debrk();

% get here on fork termination psi %
   (fkterm):
   % if its the fork we are interested in notify the user and
   cleanup, else do nothing %
      IF NOT infork THEN !debrk();
      !rfsts( infork );
      CASE r1.LH .A 377777B OF
         = 2:
            BEGIN
            IF trmcod THEN
               BEGIN  % we got fork term psi before term char psi %
               % deactivate and deassign term char psi %
                  !dti( trmcod := 0 );
                  !dic( 400000B, 1);
                  chntab[35] ← 0;
               % get back our proper tiw %
                  !stiw( 400000B, tiw := 0);
               % now get back into NLS %
                  gprnls( tmode );
               % get rid of the inferior %
                  gpkill():
               % setup to debrk to sysrtn %
                  [levtab] ← $sysrtn;
               END
            ELSE
               % get rid of the inferior %
                  gpkill():
            dismes(2, $"Tenex Subsystem Completed");
            END;
         = 3:
            BEGIN
            IF trmcod THEN
               BEGIN  % we got fork term psi before term char psi %
               % deactivate and deassign term char psi %
                  !dti( trmcod := 0 );
                  !dic( 400000B, 1);
                  chntab[35] ← 0;
               % get back our proper tiw %
                  !stiw( 400000B, tiw := 0);
               % now get back into NLS %
                  gprnls( tmode );
               % tell the user what happened %
                  gpfktrm();
               % get rid of the inferior %
```

```
                gpkill();
              % setup to debrk to sysrtn %
                [levtab] ← $sysrtn;
              END
            ELSE
              BEGIN
              % tell the user what happened %
                gpfktrm();
              % get rid of the inferior %
                gpkill();
              END;
            END;
          ENDCASE;
        !debrk();

END.

% %
```

```
% support routines for doing procedure replace and backup %
  (rplproc) % Replace procedure oldname by newname %
     PROCEDURE
        (oldname, newname);
     LOCAL oldval, newval, c;
     REF oldname, newname;

     c ← 0;
     IF NOT (oldval ← jutval(&oldname, TRUE : c)) THEN
        err($"No old procedure: error in DDT lookup.");
     IF NOT (newval ← jutval(&newname, FALSE)) THEN
        err($"No new procedure: error in DDT lookup.");
     IF oldval = newval THEN RETURN( FALSE );
     % put "old" procedure in backup table. %
        IF NOT c THEN
           IF NOT tblsearch($proctbl, 10 %proctblsz%, 2
           %proctblentsz%, 0: c) THEN
              err($"Backup Table Full...Nothing Replaced");
        [c] ← oldval;
        [ c +1] ← [ oldval];
        [oldval] ← 254B9 %jrst% + newval;
     RETURN( TRUE );
     END.

  (jutval)    % find value of name %
     PROCEDURE
        (name, mode);
     LOCAL retval, c;
     REF name;

     IF NOT ddtlookup(&name, mode: retval) THEN
        RETURN(FALSE);
     IF ([retval].accum10 # $s AND [retval].addr10 # $sysovr) THEN
        IF [retval].opcod10 # 254B %JRST% THEN RETURN(FALSE)
        ELSE % Check if breakpoint or procedure replace done already %
           IF tblsearch($proctbl, 10 %proctblsz%, 2 %proctblentsz%,
           retval:c) THEN RETURN(retval, c)
           ELSE RETURN(FALSE);
     RETURN(retval, 0);
     END.

  (bkproc)    % do procedure backup %
     PROCEDURE( name );
     LOCAL oldval, c;
     REF name;

     IF NOT (oldval ← jutval(&name, TRUE :c)) THEN RETURN(FALSE);
     IF c = 0 THEN RETURN(FALSE);
     [oldval] ← [c+1]; % Original backup instruction. %
     [c] ← 0; % Clear it in table. %
     RETURN(TRUE);
     END.

% support routines for running inferior tenex subsystems %
  (gpfktrm)   % print tenex subsytem status from within fork
  termination psi routine %
```

```
PROCEDURE:
LOCAL STRING
   tstring[500],
   strng[500];

gptxst( infork, $strng);
*tstrng* ← "
Tenex Subsystem Aborted
", *strng*;
dismes( 2, $tstrng);
RETURN;

END.

%  %
```

```
(gprnls)    % restore nls display %
   PROCEDURE
      ( ttymode      % teminal mode to be set up %
      );

   % restore subsystem name %
      !setnm( nlssbn );
   % restore terminal mode words %
      !sfmod( 100B, ttymode);
   % restore the display %
      continue ← TRUE;
      initdis();
      continue ← FALSE;
      dpset( dspallf, endfil, endfil, endfil);
   % should be done, so return %
      RETURN;

   END.

   %  %
```

```
(upgcnv)   PROCEDURE   % get user program address from name or number%
   (str);   % address of string containing name or number %
   % given the name or number of a user program in a string, look up
   the name in the string of user program names and/or the address in
   the stack of user program starting addresses %
   LOCAL pgmidx;
   REF str;
   IF str.L = empty THEN pgmidx ← 0
   ELSE IF *str* [1] IN ['0, '9] THEN pgmidx ← cvsno (&str)
   ELSE
      BEGIN
      astruc( &str );
      pgmidx ← upgskix;
      FIND SE(*upgnms*) $NP;
      WHILE NOT (FIND < ENDCHR) DO
         BEGIN
         FIND < $NP $PT;   % move to next visible %
         IF (FIND > *str* (NP/ENDCHR)) THEN EXIT LOOP
         ELSE BUMP DOWN pgmidx;
         END;
      END;
   IF NOT pgmidx IN [1, upgskix] THEN
      SIGNAL (upgerr, $"illegal user program spec");
   RETURN (upgstk[pgmidx]);
   END.

(popsym)   PROCEDURE;  % pops block of symbols from DDT's symbol table
%
   ddtpop();
   RETURN;
   END.

(marksym)  PROCEDURE   % marks DDT's symbol table %
   (blocknm);   % 0 or string containing the block name %

   LOCAL STRING str[50];
   REF blocknm;
   IF &blocknm THEN *str* ← *blocknm*
   ELSE *str* ← "LOCAL";
   astruc( $str ); % force to upper case %
   ddtmark($str);
   RETURN;
   END.

(upgjfn)       % get a jfn for parsed link data structure %
   PROCEDURE
      (adstr);          % address of parsed link data structure %
   LOCAL try, jfn;
   LOCAL TEXT POINTER tp1, tp2;
   LOCAL STRING filename[200], dirname[100];
   REF adstr;

   % ALGORITHM %
      % if a directory name is specified for the link then this
      directory is searched for the file.  if no directory name is
      specified then the following directories are searched for the
```

```
   file (in the following order):
      NETSYS
      SUBSYS
      the default directory for links for the link
      the connected directory
      the login directory
   in all cases the default extension name is "SAV". %
% RETURNS %
   % this routine will return the jfn for the found file or FALSE;
   it will generate an err for links which specify a remote hostname
   %

% get file name as specified in link (& check for local host) %
   CASE lnpfls( 0, &adstr, $filename) OF
      = lhostn: NULL;
      ENDCASE err($"Remote File Manipulations Not Implemented Yet");
% only try once if directory specified in link %
   IF adstr[ue+1] > adstr[us+1] THEN
      RETURN( lgetjfn(0, $filename, $savext, gtjprf, $lit) );
% now try various directories %
   tp1 ← adstr[fs]:  tp1[1] ← adstr[fs+1];
   tp2 ← adstr[fe]:  tp2[1] ← adstr[fe+1];
   *filename* ← tp1 tp2;
   try ← 0;
   *dirname* ← "NETSYS";
   LOOP
      CASE (jfn← lgetjfn($dirname,$filename,$savext,gtjprf,$lit)) OF
         = FALSE:
            CASE try ← try + 1 OF
               = 1:          % SUBSYS %
                 *dirname* ← *subdir*;
               = 2:          % default directory for links %
                 IF adstr[lfn] THEN gdftdir( adstr[lfn], $dirname)
                 ELSE REPEAT CASE;
               = 3:          % connected directory %
                 *dirname* ← NULL;
               = 4:          % login directory %
                 *dirname* ← *userstr*;
               ENDCASE err($lit);
         ENDCASE RETURN( jfn ):
END.

FINISH
```

ADRMNP

(MLK)ADRMNP (MLK)ADRMNP (MLK)ADRMNP (MLK)ADRMNP (MLK)ADRMNP (M
(MLK)ADRMNP (MLK)ADRMNP (MLK)ADRMNP (MLK)ADRMNP (MLK)ADRMNP (M

```
< NLS, ADRMNP.NLS;41, >, 18-SEP-74 14:43 KEV ;;;;< NLS, ADRMNP.NLS;41,
>, 18-JUN-74 13:21 KEV ;
FILE adrmnp  % 110  <rel-nls>adrmnp % % (110,)  (rel-nls,adrmnp.rel,)
%                                                                    0175
   % DOCUMENTATION %                                                 01655
      % the following is the formal description of a link that is used
      by the procedures that parse links:                            01656
         s  :=  [SP / TAB / CR / LF / EOL]  /  s (SP / TAB / CR / LF /
         EOL)                                                        01657
         link  :=  opndlm  s  body  s  clsdlm                        01658
            opndlm  :=  ( ('< / '() [comment] )  /  "--"             01659
            clsdlm  :=  ')  /  '>                                    01660
            comment  :=  ctxt "--"                                   01661
            ctxt  :=  any number of characters excluding the following:
                                                                     01662
               '- / ', / '> / ') / '< / '( / '. / '; / ': /          01663
               '! / '* / '# / '/ / '\ / '" / '' / '= / '+            01664
         body  :=  [filspc]  [dae]  [': vwspc]                       01665
         filspc  :=  [[hn ',] un ',] fn ',                           01666
         hn  :=  s hstnam s  /  NULL                                 01667
            hstnam  :=  1$48(LD/'-)                                  01668
         un  :=  s usrnam s                                          01669
            usrnam  :=  zero to 39 characters excluding the following
                                                                     01670
               ', / SP / TAB / EOL / ': / '; /                       01671
               '< / '> / '= / '← / '* / '@ / '. /                    01672
               [0B,5B] / [7B,32B] / [34B,36B] / 140B / >= 173B       01673
         fn  :=  s (filnam / filnam2) s                              01674
            filnam  :=  zero or more characters excluding the following
                                                                     01675
               ', / SP / TAB / EOL / ': /                            01676
               '< / '> / '= / '← / '@ /                              01677
               [0B,5B] / [7B,32B] / [34B,36B] / 140B / >= 173B       01678
            filnam2  :=  '= delim1 STRING delim2                     01679
            delim1  :=  CHARACTER                                    01680
            delim2  :=  the same character used for delim            01681
         dae  :=  element  /  dae element                            01682
         element  :=                                                 01683
            s  /                                                     01684
            stmntnam  /  '! stmntnam  /  '* stmntnam  /              01685
            '# marker  /                                             01686
            '/  /  '\  /                                             01687
            ('" STRING '"  /  '' CHAR) s ['= search]  /              01689
            '. pelement  /  '+ selement  /  '- selement              01690
         marker  :=                                                  01691
            a letter followed by any number of characters except the
            following:                                               01692
               SP / TAB / ') / '> / ':                               01693
         stmntnam  :=                                                01694
            LD  [LD / '- / '' / '@]  /  stmntnam  (LD / '- / '' / '@)
                                                                     01695
         search  :=  stype [sdomain]  /  sdomain [stype]             01696
         stype  :=  [NUMBER] ('C  /  'W)                             01697
            (can be upper or lower case letters)                     01698
         sdomain  :=  [NUMBER] 'S                                    01699
            (can be upper or lower case letters)                     01700
```

```
selement  :=  [NUMBER] struc  /  selement [NUMBER] struc   01702
struc  :=  'C  /  'E  /  'F  /  'I  /  'L  /  'N  /  'V  /  'W
                                                           01703
   (can be upper or lower case letters)                    01704
pelement  :=  [NUMBER] pos  /  pelement [NUMBER] pos       01705
pos  :=                                                    01706
   'B  /  'C  /  'D  /  'E  /  "FR"  /  'H  /  'L  /       01707
   'N  /  'O  /  'P  /  'R  /  'S  /  'T  /  'U  /  'W     01708
      (can be upper or lower case letters)                 01709
vwspc  :=  [viewspec]  /  vwspc viewspec                   01710
viewspec  :=                                               01711
    s  /                                                   01712
   'a  /  'b  /  'c  /  'd  /  'e  /  'f  /  'g  /  'h  /  01713
   'i  /  'j  /  'k  /  'l  /  'm  /  'n  /  'o  /  'p  /  01714
   'q  /  'r  /  's  /  't  /  'u  /  'v  /  'w  /  'x  /  01715
   'y  /  'z  /  'A  /  'B  /  'C  /  'D  /  'G  /  'H  /  01716
   'I  /  'J  /  'K  /  'L  /  'O  /  'P  /  filter       01717
filter  :=  '; CONTENT-PATTERN ';                          01718
%                                                          01720
```

```
REGISTER r1 = 1, r2 = 2, r3 = 3;                                        01231
% Link field extraction. %                                             05974
   (lnkpspc)  %extract fields from link and setup link data block%
                                                                        05502
      PROCEDURE                                                         05503
         (lnknum,              %link number%                            05504
         tptadr,               %t-ptr from which to start link search%
                                                                        05505
         usrstg,               %string for directory name%              05506
         fnmstg,               %string for file name%                   05507
         addexpstg,            %string for address expression%          05508
         vspstg,               %string for viewspecs%                   05509
         adstr);               % addres of data block to be filled %
                                                                        05510
      LOCAL                                                             05511
         ind,                                                           05512
         rhstn;      % host number %                                     05513
      LOCAL TEXT POINTER                                                05514
         tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8;                        05515
      REF tptadr, usrstg, fnmstg, addexpstg, vspstg, adstr;             05516
                                                                        05517
      % find and parse the right link and get the file name %            05518
         IF &tptadr THEN                                                05519
            BEGIN                                                       05520
            tp8 ← tptadr;  tp8[1] ← tptadr[1];                          05521
            ind ← 1;                                                    05522
            DO                                                          05523
               BEGIN                                                    05524
               rhstn ← lnbfls( $tp8, &adstr, &fnmstg);                  05525
               CASE ind OF                                              05526
                  = 1: NULL;                                            05527
                  ENDCASE                                               05528
                     IF adstr[ls+1] <= tp8[1] THEN                      05529
                        BEGIN                                           05530
                        SIGNAL( lnk5err, $"Illegal Link:
                        Left Delimiter Not Found");                     05531
                        END;                                            05532
               tp7 ← adstr[ls];  tp7[1] ← adstr[ls+1];                  05533
               tp8 ← adstr[le];  tp8[1] ← adstr[le+1];                  05534
               BUMP ind;                                                05535
               END UNTIL ind > lnknum;                                  05536
            END                                                         05537
         ELSE                                                           05538
            BEGIN                                                       05539
            rhstn ← lnbfls( 0, &adstr, &fnmstg );                       05540
            tp7 ← adstr[ls];  tp7[1] ← adstr[ls+1];                     05541
            END;                                                        05542
      % set up other strings %                                          05543
         tp1 ← adstr[us];  tp1[1] ← adstr[us+1];                        05544
         tp2 ← adstr[ue];  tp2[1] ← adstr[ue+1];                        05545
         *usrstg* ← + tp1 tp2; % directory %                            05546
         tp3 ← adstr[ds];  tp3[1] ← adstr[ds+1];                        05547
         tp4 ← adstr[de];  tp4[1] ← adstr[de+1];                        05548
         *addexpstg* ← tp3 tp4; % address expression %                  05549
         tp5 ← adstr[vb];  tp5[1] ← adstr[vb+1];                        05550
         tp6 ← adstr[ve];  tp6[1] ← adstr[ve+1];                        05551
```

```
   *vspstg* + tp5 tp6; % view %                                    05552
RETURN( rhstn );                                                   05553
END.
                                                                   05554
%  %                                                              05555
```

```
(lnkfls)          % create file name string from link parse block%
                                                                    03409
                                                                    03410
   PROCEDURE
      (stp,       % FALSE or t-ptr for use for parsing a link %
                                                                    03411
      adstr,      % address of parsed link data structure or zero
      %
                                                                    03412
      astr);      % address of string to get file name %          03413
                                                                    03414
   LOCAL
      i,          % index for copying data structures %           03415
      rhstn,      % cell to get remote host number %              03416
      ldstr[40];          % local data structure for link parsing
      %
                                                                    03417
   LOCAL TEXT POINTER                                               03418
      h1, h2, u1, u2, f1, f2;                                       03419
   LOCAL STRING                                                     03420
      locstr[200];                                                  03421
   REF stp, adstr, astr;                                            03422
                                                                    03423
   % RETURNS %                                                     03424
      % will generate error if invalid hostname field in link.
      Otherwise, will return filled in string that can be used
      for gtjfn and will return hostnumber (number for local host
      if none specified in the link %                               03425
                                                                    03426
   % ALGORITHM %                                                   03427
      % for local hosts the following is how the file string is
      built up:                                                     03428
         if a user name is specified (that doesn't end with a
         control-f or an altmode) then the file name is the
         concatenation of:                                          03429
            '< specified-user-name '> specified-file-name           03430
               if the file name is not specified, it is set to
               the file currently loaded (or generates an error
               for typed in links)                                  03431
         if a user name is specified (that ends with a control-f
         or an altmode) then the file name is the concatenation
         of:                                                        03432
            '< specified-user-name specified-file-name              03433
               if the file name is not specified, it is set to
               the file currently loaded (or generates an error
               for typed in links)                                  03434
         if no user name is specified for a typed in link, then
         the file name string is merely the specified file name
         (or NULL if no file name specified)                        03435
         if no user name is specified for a bugged link, then the
         file name string consists of the concatenation of:         03436
            '< default-directory '> specified-file-name             03437
               if the file name is not specified, it is set to
               the file currently loaded                            03438
      for remote hosts, if a user name is specified, then the
      same algorithm is used as is used for the local host.         03439
      for remote hosts without a user name, the returned string
      is the specified file name (or generates an error if no
      file name specified for a typed in link)                      03440
      %                                                             03441
```

```
                                                                    03442
% parse link or copy data structure as necessary %                  03443
   IF &stp THEN                                                     03444
      BEGIN                                                         03445
      lnkprs( &stp, $ldstr);                                        03446
      IF &adstr THEN                                                03447
         FOR i ← 1 UP UNTIL > lnkdsl DO adstr[i-1] ←
         ldstr[i-1];                                                03448
      END                                                           03449
   ELSE                                                             03450
      IF &adstr THEN                                                03451
         FOR i ← 1 UP UNTIL > lnkdsl DO ldstr[i-1] ←
         adstr[i-1]                                                 03452
      ELSE err( $"NLS System Error: Illegal Call to LNBFLS");
                                                                    03453
% initialize text pointers %                                       03454
   h1 ← ldstr[hs];  h1[1] ← ldstr[hs+1];                            03455
   h2 ← ldstr[he];  h2[1] ← ldstr[he+1];                            03456
   u1 ← ldstr[us];  u1[1] ← ldstr[us+1];                            03457
   u2 ← ldstr[ue];  u2[1] ← ldstr[ue+1];                            03458
   f1 ← ldstr[fs];  f1[1] ← ldstr[fs+1];                            03459
   f2 ← ldstr[fe];  f2[1] ← ldstr[fe+1];                            03460
% pick up any specified user name %                                03461
   *locstr* ← + u1 u2;                                              03462
% act according to whether its a local host or not %               03463
   CASE (rhstn ← gtnstn( $h1, $h2 ) ) OF                            03464
      < 0:  err($"invalid host name");                              03465
      = lhostn:       % local hosts %                               03466
         (lcl):                                                     03467
         CASE locstr.L OF                                           03468
            = 0:      % no user name specified %                    03469
               BEGIN                                                03470
               *locstr* ← f1 f2;                                    03471
               CASE ldstr[lfn] OF                                   03472
                  = 0:          % typed in link %                   03473
                     IF locstr.L THEN *astr* ← *locstr*             03474
                     ELSE *astr* ← NULL;                            03475
                  ENDCASE       % bugged link %                    03476
                     BEGIN                                          03477
                     gdftdir( ldstr[lfn], &astr);                   03478
                     *astr* ← '<, *astr*, '>;                       03479
                     IF locstr.L THEN *astr* ← *astr*,
                     *locstr*                                       03480
                     ELSE lnbafn( &astr, ldstr[lfn] );              03481
                     END;                                           03482
               END;                                                 03483
            ENDCASE   % user name specified %                       03484
               BEGIN                                                03485
               CASE *locstr*[locstr.L] OF                           03486
                  = '<↑F>, = '<ESC>:                                03487
                     *astr* ← '<, *locstr*;                         03488
                  ENDCASE                                           03489
                     *astr* ← '<, *locstr*, '>;                     03490
               *locstr* ← f1 f2;                                    03491
               CASE locstr.L OF                                     03492
                  = 0:                                              03493
```

```
                        IF ldstr[lfn] THEN lnbafn( &astr,
                        ldstr[lfn] )                                    03494
                        ELSE err($"Illegal Link");                      03495
                     ENDCASE                                            03496
                        *astr* ← *astr*, *locstr*;                      03497
                  END;                                                  03498
      ENDCASE          % remote hosts %                                 03499
         IF locstr.L THEN GOTO lc1                                      03500
         ELSE                                                           03501
            BEGIN                                                       03502
            *astr* ← f1 f2;                                             03503
            IF NOT astr.L THEN                                          03504
               IF NOT ldstr/lfn/ THEN err($"Illegal Link")
                                                                        03505
               ELSE                                                     03506
                  lnbafn( &astr, ldstr[lfn] );                          03507
            END;                                                        03508
% now return %                                                         03509
   RETURN( rhstn );                                                     03510
END.
                                                                        03511
%  %                                                                   03512
```

```
(lnbafn)          % append file name string to passed string %
                                                                    03513
   PROCEDURE                                                        03514
      (astr,      % string to be appended to %                       03515
      fileno);    % file number for file name string %              03516
   LOCAL                                                            03517
      ldstr[40];          % data structure to parse link %           03518
   LOCAL TEXT POINTER                                               03519
      tp1, tp2, tp3;                                                03520
   LOCAL STRING locstr[200];                                        03521
   REF astr;                                                        03522
                                                                    03523
   % get the file name %                                           03524
      filnam( fileno, $locstr);                                     03525
   % now parse the link %                                          03526
      FIND SF(*locstr*) ↑tp3;                                       03527
      lnkprs( $tp3, $ldstr);                                        03528
      tp1 ← ldstr[fs];  tp1[1] ← ldstr[fs+1];                       03529
      tp2 ← ldstr[fe];  tp2[1] ← ldstr[fe+1];                       03530
   % do the appending and return %                                 03531
      *astr* ← *astr*, tp1 tp2;                                     03532
      RETURN;                                                       03533
   END.
                                                                    03534
   %  %                                                            03535
```

```
(lnkprs)          % detailed link parser %                                  04929
                                                                           04930
   PROCEDURE                                                               04931
      (stp,       % address of text pointer to start scan %                04932
      adstr       % address of  linkparams  data strcture %                04933
      );                                                                   04934
   LOCAL rflag;                                                            04935
   LOCAL TEXT POINTER tp1, tp2;                                            04936
   REF stp, adstr;                                                         04937

   % the following signals (maintained in CONST) can be generated
   while parsing a link:                                                   04938
      lnk1err -                                                            04939
         Illegal Link Syntax:
         ENDCHR Before Closing "                                           04940
      lnk2err -                                                            04941
         Illegal Link Syntax:
         ENDCHR Before CH After '                                          04942
      lnk3err -                                                            04943
         Illegal Link Syntax:
         Illegal Statement Name or Number After ! or *                     04944
      lnk4err -                                                            04945
         Illegal Link Syntax:
         Illegal DAE Element                                               04946
      lnk5err -                                                            04947
         Illegal Link:
         Left Delimeter Not Found                                          04948
      lnk6err -                                                            04949
         Illegal Link Syntax or Semantic:
         Missing Right Delimeter or Bad Viewspecs                          04950
      lnk7err -                                                            04951
         Illegal Link Syntax:
         Missing Right Delimeter                                           04952
      lnk8err -                                                            04953
         Illegal Link Syntax:
         Illegal Marker After #                                            04954
      lnk9err -                                                            04955
         Illegal Link Syntax:
         ENDCHR Before ; In Filter                                         04956
      %                                                                    04957
                                                                           04958
   % initialize %                                                          04959
      rflag ← TRUE;                                                        04960
   % setup working t-ptr %                                                 04961
      FIND stp ↑tp1 ↑tp2;                                                  04962
      IF FIND > '- THEN FIND < $'- ↑tp1 ↑tp2;                              04963
   % setup default directory file number %                                 04964
      adstr[lfn] ← stp.stfile;                                             04965
   % try different starting points if errors %                             04966
      ON SIGNAL                                                            04967
         # lnk5err:                                                        04968
            IF rflag AND (adstr[ls+1] >= tp1[1]) THEN                      04969
               BEGIN                                                       04970
               tp1[1] ← adstr[cs+1];                                       04971
               GOTO lnkps1;                                                04972
               END                                                         04973
            ELSE                                                           04974
```

```
                BEGIN                                                   04975
                rflag ← FALSE;                                          04976
                tpl[1] ← MIN(tpl[1], tp2[1], adstr[ls+1]);              04977
                IF FIND tpl < ['( / '< / "--"] ↑tpl >                   04978
                   THEN GOTO lnkps1;                                    04979
                END;                                                    04980
        ELSE;                                                           04981
(lnkps1):                                                               04982
% find start of the link %                                              04983
   lnkstr( $tpl, &adstr);                                               04984
% find an optional comment %                                           04985
   lnkcom( &adstr );                                                    04986
% find the body of the link %                                          04987
   lnkbdy( &adstr );                                                    04988
% find the right delimeter %                                           04989
   lnkend( &adstr );                                                    04990
% done, so return %                                                    04991
   RETURN;                                                              04992
                                                                        04993
END.                                                                    04994
% %                                                                     04995
```

```
(lnkstr)          % find the start of a link %                          091
                                                                       095
   PROCEDURE
      (stp,       % address of text pointer to start scanninfg
      from %                                                           092
      adstr       % address of  linkparams  data strcture %            093
      );                                                               094
   LOCAL lnktyp;                                                      01905
   LOCAL TEXT POINTER tpl;                                             0197
   REF stp, adstr;                                                     096
                                                                       097
   % ABNORMAL RETURNS %                                               01321
      % can generate the following signal:                            01322
         lnkSerr:                                                     01323
            Illegal Link:
            Left Delimeter Not Found                                   01324
      %                                                                01325
                                                                       01326
   % position to scan starting text pointer %                          098
      FIND stp > ↑tpl;                                                 099
      IF NOT stp.stastr THEN                                          04924
         BEGIN                                                        04925
         tpl[1] ← MAX( tpl[1], fchtxt(getsdb(stp)) );                 04926
         FIND tpl >;                                                  04927
         END;                                                         04928
   % initially normal type link %                                     0158
      lnktyp ← FALSE;                                                  0159
   LOOP                                                                0105
      CASE READC OF                                                    0100
         = '(, = '<:                                                   0101
            BEGIN                                                      0170
            FIND ↑tpl ←tpl;                                            0168
            EXIT LOOP;                                                 0171
            END;                                                       0172
         = '-:                                                         0106
            IF FIND 1$'- < 20B ↑tpl THEN                               0162
               BEGIN                                                   0163
               lnktyp ← TRUE;                                          0164
               EXIT LOOP;                                              0165
               END;                                                    0166
         = ENDCHR, = '), = '>:                                         0107
            BEGIN                                                      0199
            FIND stp <;                                                0200
            LOOP                                                       0202
               CASE READC OF                                           0203
                  = '(, = '<:                                          0204
                     BEGIN                                             0205
                     FIND ↑tpl;                                        0206
                     EXIT LOOP 2;                                      0207
                     END;                                              0208
                  = '-:                                                0209
                     IF FIND '- ↑tpl THEN                              0210
                        BEGIN                                          0211
                        lnktyp ← TRUE;                                 0212
                        EXIT LOOP 2;                                   0213
                        END;                                           0214
                  = ENDCHR:                                           01336
```

```
                    BEGIN                                                   01311
                    tpl[1] ← 0;                                             01312
                    EXIT LOOP 2;                                            01313
                    END;                                                    01314
                ENDCASE;                                                     0219
            END;                                                             0201
        ENDCASE;                                                             0108
IF (NOT tpl[1]) OR (NOT stp.stastr) THEN                                    01947
    IF (NOT tpl[1]) OR (tpl[1]<fchtxt(getsdb(stp))) THEN                     0221
        BEGIN                                                               01315
        SIGNAL( lnkSerr, $"Illegal Link:
        Left Delimiter Not Found");                                         01318
        END;                                                                01319
adstr[ls] ← tpl;  adstr[ls+1] ← tpl[1];                                      0198
% initialize to no comment %                                               01906
    adstr[cs] ← adstr[ce] ← adstr[ls];                                      01907
    adstr[cs+1] ← adstr[ce+1] ← adstr[ls+1]+1;                              01908
% no comment if link starts with "--" %                                     01909
    IF lnktyp THEN                                                          01910
        BEGIN                                                               01911
        % make comment live after "--" %                                    01912
            adstr[cs+1] ← adstr[ce+1] ← adstr[ls+1] + 2;                    01913
        END;                                                                01915
RETURN;                                                                      0835
END.

                                                                             0176
% %                                                                         0252
```

```
(lnkcom)          % find the optional comment in a link %        0226
                                                                0227
   PROCEDURE                                                    0228
      (adstr      % address of  linkparams  data strcture %     0229
      );                                                        0238
   LOCAL TEXT POINTER tpl;                                      0230
   REF adstr;                                                   0231
                                                                0257
   % initialize %                                               0234
      tpl ← adstr[cs];                                          0235
      tpl[1] ← adstr[cs+1];                                     0239
   % no comment if link starts with "--" %                      0232
      IF (adstr[ce+1] - adstr[cs+1]) = 2 THEN RETURN;
   % setup to scan forward from start of link looking for comment
   %                                                            0240
      FIND tpl >;                                               0243
   % scan until "--" or any other link delimeter %             0244
      LOOP                                                      0245
         CASE READC OF                                          0246
            = '-:                                               0247
               IF FIND 1s'- ↑tpl THEN                           0261
                  BEGIN                                         0262
                  adstr[ce] ← tpl;                              0263
                  adstr[ce+1] ← tpl[1];                         0264
                  RETURN;                                       0265
                  END                                           0266
               ELSE RETURN;                                     0267
            = ENDCHR,                                           0255
            = ',, = '>, = '), = '<, = '(, = '.,                 0248
            = ';, = ':, = '↓, = '*, = '#, = '/,                 0253
            = '\, = '", = '', = '=, = '+:                       0254
               RETURN;                                          0260
            ENDCASE;                                            0256
                                                                0270
   END.                                                         027
   % %                                                          0249
```

```
(lnkbdy)          % find the body of a link %                    01573
                                                                 01574
   PROCEDURE                                                     01575
      (adstr      % address of  linkparams  data strcture %      01576
      );                                                         01577
   LOCAL TEXT POINTER stp;                                       01578
   REF adstr;                                                    01579
                                                                 01580
   % find the optional filspc part of the link %                 01581
      lnkfpc( &adstr, $stp );                                    01582
   % find the optional filspc part of the link %                 01583
      lnkdae( &adstr, $stp );                                    01584
   % find the optional vwspc part of the link %                  01585
      lnkvws( &adstr, $stp );                                    01586
   % done, so return %                                          01587
      RETURN;                                                    01588

   END.                                                          01589

   %  %                                                         01590
```

```
(lnkfpc)          % find the optional filspc of a link %          0377
                                                                  0378
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %      0379
      tpl         % address of text pointer to get end of filspc %
                                                                  0999
      );                                                          0382
   REF adstr, tpl;                                                0380
                                                                  0381
   % find an optional hostname %                                  0272
      lnkhst( &adstr, &tpl );                                     0273
   % find an optional username %                                  0332
      lnkusr( &adstr, &tpl );                                     0333
   % find an optional filename %                                  0459
      lnkfil( &adstr, &tpl );                                     0460
   % done so return %                                            0391
      RETURN;                                                     0392
                                                                  0393
   END.
                                                                  0394
   %  %                                                          0395
```

```
(lnkhst)          % find the optional hostname in a link %      0334
   PROCEDURE                                                   0335
      (adstr,     % address of  linkparams  data strcture %    0336
      stp         % text pointer gets end of hostname field % 01002
      );                                                       0337
   LOCAL TEXT POINTER tp1, tp2;                                0338
   REF adstr, stp;                                             0339
                                                               0345
   % initialize to no hostname %                               0346
      adstr[hs] ← adstr[he] ← tp1 ← adstr[ce];                 0347
      adstr[hs+1] ← adstr[he+1] ← tp1[1] ← adstr[ce+1];        0348
   % setup scan forward from end of comment (skip spaces or tabs)
   %                                                           0349
      FIND tp1 > $(SP/TAB/CR/LF/EOL) ↑tp1 ↑stp;                0350
   % hostname is 0-48 letters, digits, or minus signs, followed
   by any number of spaces or tabs, and terminated by a comma %
                                                               0351
      IF FIND $48(LD/'-) ↑tp2 $(SP/TAB/CR/LF/EOL) ', ↑stp THEN
                                                               0352
         BEGIN                                                 0353
         adstr[hs] ← tp1;                                      0354
         adstr[hs+1] ← tp1[1];                                 0355
         adstr[he] ← tp2;                                      0356
         adstr[he+1] ← tp2[1];                                 0357
         RETURN;                                               0359
         END                                                   0360
      ELSE RETURN;                                             0361
                                                               0362
   END.
                                                               0363
   % %                                                        0364
```

```
(lnkusr)          % find the optional username in a link %      0461
                                                               0462
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %     0463
      stp         % text ptr to start scan; gets end of username
      field %                                                  0464
      );                                                       0466
   LOCAL                                                       0962
      i  % character count index %                             0963
      ;                                                        0964
   LOCAL TEXT POINTER tp1, tp2;                                0467
   REF adstr, stp;                                             0468
                                                               0475
   % initialize to no username %                               0476
      adstr[us] ← adstr[ue] ← adstr[he];                       0477
      adstr[us+1] ← adstr[ue+1] ← adstr[he+1];                 0478
   % setup scan forward from end of hostname (skip spaces & tabs)
   %                                                           0479
      FIND stp > $(SP/TAB/CR/LF/EOL) ↑tp1 ↑stp;                0480
   % find username if it exists, else make hostname username and
   make hostname null %                                        0484
      FOR i ← 1 UP UNTIL > 39 DO                               0485
        CASE READC OF                                          0487
           % illegal username characters %                     0488
              = ',, = SP, = TAB, = CR, = LF, = EOL:            0489
                 BEGIN                                         0490
                 IF NOT FIND < CH ↑tp2 > $(SP/TAB/CR/LF/EOL) ',
                 ↑stp THEN                                     0491
                    GOTO notusr;                               0492
                 adstr[us] ← tp1;  adstr[us+1] ← tp1[1];       0493
                 adstr[ue] ← tp2;  adstr[ue+1] ← tp2[1];       0494
                 RETURN;                                      01003
                 END;                                          0506
              = ':, = ';, = '<, = '>, = '=, = '←, = '*, = '@, =
              '.,                                              0507
              = ENDCHR,                                        0508
              IN [0B,5B]          % ↑@ - ↑E %,                  0509
              IN [78,32B]         % ↑G - ↑Z %,                  0510
              IN [34B,36B]        % ↑J - ↑↑ %,                  0511
              = 140B, >= 173B:                                 0512
                 (notusr):        BEGIN                        0513
                 % make username field = hostname field %      0514
                    adstr[us] ← adstr[hs];                     0515
                    adstr[us+1] ← adstr[hs+1];                 0516
                    adstr[ue] ← adstr[he];                     0517
                    adstr[ue+1] ← adstr[he+1];                 0518
                 % make hostname field null %                  0519
                    adstr[he] ← adstr[hs];                     0520
                    adstr[he+1] ← adstr[hs+1];                 0521
                 RETURN;                                       0522
                 END;                                          0523
           ENDCASE;                                            0524
      % more than 39 characters is not a user name %           0960
        GOTO notusr;                                           0961
                                                               0525
   END.
                                                               0526
```

% %

```
(lnkfil)          % find the optional filename in a link %      0274
   PROCEDURE                                                   0275
      (adstr,     % address of  linkparams  data strcture %    0276
      stp         % text ptr to start scan; gets ent of filename
      field %                                                  0366
      );                                                       0277
   LOCAL                                                      01026
      char        % delimeter character %                     01027
      ;                                                       01028
   LOCAL TEXT POINTER tp1, tp2;                               0278
   REF adstr, stp;                                            0279
                                                              0320
   % initialize to no filename %                              0281
      adstr[fs] ← adstr[fe] ← adstr[ue];                      0282
      adstr[fs+1] ← adstr[fe+1] ← adstr[ue+1];                0283
   % setup scan forward from end of username (skip spaces & tabs)
   %                                                          0291
      FIND stp > $(SP/TAB/CR/LF/EOL) ↑tp1 ↑stp;               0294
   % find filename if it exists, else make username filename and
   make username hostname and make hostname null %            0396
      CASE READC OF                                          01004
         = ':=:                                              01005
            CASE char ← READC OF                             01009
               = ENDCHR:  GOTO notfil;                       01010
               ENDCASE                                       01011
                  BEGIN                                      01017
                  FIND ↑tp1;                                 01018
                  LOOP                                       01012
                     CASE READC OF                           01013
                        = ENDCHR:  GOTO notfil;              01014
                        = char:                              01015
                           IF FIND < CH ↑tp2 > CH
                           $(SP/TAB/CR/LF/EOL) ', ↑stp       01020
                              THEN GOTO isfil                01022
                           ELSE GOTO notfil;                 01021
                        ENDCASE;                             01016
                  END;                                       01019
         = ENDCHR: GOTO notfil;                              03654
         ENDCASE                                             01006
            BEGIN                                            01333
            FIND < CH >;                                     01335
            LOOP                                              0398
               CASE READC OF                                  0402
                  % illegal filename characters %             0418
                     = ',, = SP, = TAB, = CR, = LF, = EOL:    0436
                        BEGIN                                 0438
                        IF NOT FIND < CH ↑tp2 >
                        $(SP/TAB/CR/LF/EOL) ', ↑stp           0437
                           THEN GOTO notfil;                 01007
                        (isfil):                             01023
                           adstr[fs] ← tp1;  adstr[fs+1] ←
                           tp1[1];                            0440
                           adstr[fe] ← tp2;  adstr[fe+1] ←
                           tp2[1];                            0441
                           RETURN;                           01008
                        END;                                  0445
```

```
                    = ':, = '<, = '>, = '=, = '←, = '@,                    0413
                    = ENDCHR,                                              0419
                    IN [0B,5B]  % ↑@ - ↑E %,                               0414
                    IN [7B,32B]          % ↑G - ↑Z %,                       0415
                    IN [34B,36B]         % ↑] - ↑↑ %,                       0416
                    = 140B, >= 173B:                                       0417
                       (notfil):         BEGIN                             0422
                       % make filename field = username field %
                                                                           0451
                          adstr[fs] ← adstr[us];                           0423
                          adstr[fs+1] ← adstr[us+1];                       0452
                          adstr[fe] ← adstr[ue];                           0424
                          adstr[fe+1] ← adstr[ue+1];                       0453
                       % make username field = hostname field %
                                                                           0528
                          adstr[us] ← adstr[hs];                           0529
                          adstr[us+1] ← adstr[hs+1];                       0530
                          adstr[ue] ← adstr[he];                           0531
                          adstr[ue+1] ← adstr[he+1];                       0532
                       % make hostname field null %                       0454
                          adstr[he] ← adstr[hs];                           0425
                          adstr[he+1] ← adstr[ns+1];                       0455
                       RETURN;                                             0428
                       END;                                                0429
                 ENDCASE;                                                  0420
           END;                                                           01334
                                                                           0314
END.
                                                                           0315
% %                                                                       0316
```

```
(lnkdae)          % find the optional dae in a link %                    01339
                                                                        01340
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %             01341
      stp         % text pointer to start scan; gets end of dae %
                                                                        01342
      );                                                                01343
   LOCAL                                                                01344
      lnkpar[29]          % additionl link params block for
      recursion %                                                       01345
                                                                        01346
      ;
   LOCAL TEXT POINTER tp1, tp2, tp3;                                    01347
   REF adstr, stp;                                                      01351
                                                                        01352
   % ABNORMAL RETURNS %                                                01353
      % this procedure can generate the following signals:             01354
         lnk2err:                                                       01357
            Illegal Link Syntax:
            ENDCHR Before CH After '                                    01358
      %                                                                 01363
                                                                        01364
   % initialize to no dae %                                            01365
      (daeprs):                                                         04486
      adstr[ds] ← adstr[de] ← adstr[fe];                                01366
      adstr[ds+1] ← adstr[de+1] ← adstr[fe+1];                          01367
   % setup scan forward from end of filename (skip spaces & tabs)
   %                                                                    01368
      FIND stp > $(SP/TAB/CR/LF/EOL) ↑tp1 ↑stp;                         01369
   % find dae if it exists %                                           01370
      LOOP                                                              01371
         CASE READC OF                                                  01372
            = '::                                                       01732
               BEGIN                                                    01733
               FIND ↑stp < CH $(SP/TAB/CR/LF/EOL) ↑tp2;                 01734
               (lkgdae):                                                01735
                  IF tp2[1] < tp1[1] THEN tp2[1] ← tp1[1];              01736
                  adstr[ds] ← tp1;  adstr[ds+1] ← tp1[1];               01737
                  adstr[de] ← tp2;  adstr[de+1] ← tp2[1];               01738
                  RETURN;                                               01739
               END;                                                     01740
            = '), = '>:                                                 01741
               BEGIN                                                    01742
               FIND < CH $(SP/TAB/CR/LF/EOL) ↑tp2 ↑stp;                 01743
               GOTO lkgdae;                                             01744
               END;                                                     01745
            = '":                                                       01753
               LOOP                                                     01754
                  CASE READC OF                                         01755
                     = '", = ENDCHR:  EXIT LOOP;     .                  01762
                     ENDCASE;                                           01781
            = ':                                                        01782
               CASE READC OF                                            01783
                  = ENDCHR:                                             01784
                     BEGIN                                              01785
                     SIGNAL( lnk" err, $"Illegal Link Syntax:
                     ENDCHR Before CH After '");                        01788
                     END;                                               01789
```

```
                ENDCASE;                                                01790
        = '(, = '<, ='., = ENDCHR: % some common bad chrs %
                                                                        04467
            IF adstr[fe+1] > adstr[fs+1] THEN  % backup %               04920
                BEGIN                                                   04921
                % make filename field = username field %                04470
                   adstr[fs] ← adstr[us];                               04471
                   adstr[fs+1] ← adstr[us+1];                           04472
                   stp ← adstr[fe] ← adstr[ue];                         04473
                   stp[1] ← adstr[fe+1] ← adstr[ue+1];                  04474
                % make username field = hostname field %                04475
                   adstr[us] ← adstr[hs];                               04476
                   adstr[us+1] ← adstr[hs+1];                           04477
                   adstr[ue] ← adstr[he];                               04478
                   adstr[ue+1] ← adstr[he+1];                           04479
                % make hostname field null %                           04480
                   adstr[he] ← adstr[hs];                               04481
                   adstr[he+1] ← adstr[hs+1];                           04482
                % now reparse the dae %                                04483
                   FIND stp > $(SP/TAB/CR/LF/EOL) ', ↑stp;              04487
                   GOTO daeprs;                                         04485
                END                                                     04923
            ELSE RETURN;                                                04922
        ENDCASE;                                                        01894
                                                                        01537
END.
                                                                        01538
% %                                                                    01539
```

```
(lnkvws)          % find the optional vwspc in a link %          0871
   PROCEDURE                                                    0872
      (adstr,     % address of  linkparams  data strcture %     0873
      stp         % text pointer to start scan; gets end of dae %
                                                                01001
      );                                                        0876
   LOCAL TEXT POINTER tp1, tp2;                                 0877
   REF adstr, stp;                                              0878
                                                                01272
   % ABNORMAL RETURNS %                                         01549
      % this routine can generate the following signal:         01550
         lnk6err:                                               01551
            Illegal Link Syntax or Semantic:
            Missing Right Delimeter or Bad Viewspecs            01552
         lnk9err:                                               01642
            Illegal Link Syntax:
            ENDCHR Before ; In Filter                           01643
      %                                                         01555
                                                                01554
   % initialize to no vwspc %                                   0886
      adstr[vb] ← adstr[ve] ← adstr[de];                        0887
      adstr[vb+1] ← adstr[ve+1] ← adstr[de+1];                  0888
   % setup scan forward from end of dae (skip spaces & tabs) %
                                                                0889
      FIND stp > $(SP/TAB) ↑tp1 ↑stp;                           0890
   % find vwspc if it exists %                                  0894
      LOOP                                                      01030
         CASE READC OF                                          01031
            =SP, =TAB, =CR, =LF, =EOL, IN['a,'z], IN['A,'D],
            IN['G,'L], IN['O,'P]:                               01540
               NULL;                                            01564
            = '), ='>, = ENDCHR:                                01541
               BEGIN                                            01565
               FIND < CR $(SP/TAB/CR/LF/EOL) ↑tp2;              01566
               IF tp2[1] < tp1[1] THEN tp2[1] ← tp1[1];         01646
               adstr[vb] ← tp1;  adstr[vb+1] ← tp1[1];          01567
               adstr[ve] ← tp2;  adstr[ve+1] ← tp2[1];          01568
               RETURN;                                          01569
               END;                                             01570
            = ';:                                               01542
               LOOP                                             01558
                  CASE READC OF                                 01559
                     = ';:  EXIT LOOP;                          01560
                     = '':                                      01628
                        READC;                                  01629
                     = '":                                      01630
                        LOOP                                    01632
                           CASE READC OF                        01633
                              = '": EXIT LOOP;                  01634
                              = ENDCHR: EXIT LOOP;              01635
                              ENDCASE;                          01636
                     = ENDCHR:                                  01561
                        BEGIN                                   01637
                        SIGNAL( lnk9err, $"Illegal Link Syntax:
                        ENDCHR Before ; In Filter");            01640
                        END;                                    01641
```

```
                    ENDCASE;                                              01562
                ENDCASE                                                   01543
                    BEGIN                                                 01544
                    SIGNAL( lnkóerr, $"Illegal Link Syntax or
                    Semantics:
                    Missing Right Delimiter or Bad Viewspecs");           01547
                    END;                                                  01548
                                                                           0934
        END.
                                                                           0935
        % %                                                                0936
```

```
(lnkend)          % find the right delimeter of a link %          0838
  . PROCEDURE                                                    0839
      (adstr      % address of  linkparams  data strcture %      0840
      );                                                         0841
    LOCAL TEXT POINTER tpl;                                     01000
    REF adstr;                                                   0847
                                                                 0848
    % ABNORMAL RETURNS %                                        01601
      % this routine can generate the following signals:        01602
         lnk7err:                                               01603
            Illegal Link Syntax:
            Missing Right Delimeter                             01604
      %                                                         01605
                                                                01606
    % setup to scan from end of viewspecs %                     01591
      tpl ← adstr[ve];  tpl[1] ← adstr[ve+1];                   01592
    % find the end of the link %                                01593
      IF NOT FIND tpl > $(SP/TAB/CR/LF/EOL) (')'/'>) ↑tpl THEN
                                                                01594
         BEGIN                                                  01595
         SIGNAL( lnk7err, $"Illegal Link Syntax:
         Missing Right Delimiter");                             01598
         END;                                                   01599
    % now set up the text pointer in the linkparams block %     01609
      adstr[le] ← tpl;  adstr[le+1] ← tpl[1];                   01610
    % done, so return %                                         0842
      RETURN;                                                    0843
                                                                 0844
    END.
                                                                 0845
    %  %                                                        0846
```

```
%Address Expression evaluation%                                         01948
   (caddexp)        %Core NLS Address Expression evaluation Command%
                                                                        05556
      %given two text pointers to a text string, a pointer to a
      display area record (for viewspecs and return rings), and a
      text pointer into a file, this routine will evaluate the
      expression relative to the text pointer and update the text
      pointer.  It will also return updated viewspec words,
      content-analyzer-program, and user-seqgen program addresses.
      it will also return 0 or the address of the relevant statement
      return ring if the last thing done is a .fr  %                    05557
                                                                        05558
      PROCEDURE(t1, t2, da, ptr);                                       05559
      LOCAL                                                             05560
         savslh,     % save value of slashflg %                         05561
         rhstn,      % host number from links %                         05562
         fnlsls,     % for doing final / %                              05563
         sdomain,    % domain for content searches %                    05564
         vs1,vs2,    % saved viewspecs for errors %                     05565
         cacode,     % saved address of content analyzer program %
                                                                        05566
         useqgen,    % saved address of sequence generator program %
                                                                        05567
         proc,                                                          05568
         dir,        % direction for scan relations %                   05569
         char,       % for main loop dae parsing %                      05570
         count,      % for scan and positional relationships %          05571
         scnbck,                                                        05572
         sring,      % adr stmnt return ring for file return, etc. %
                                                                        05573
         rsrng,      % adr stmnt ret ring to be returned %              05977
         fname,      % adr file return fing for file return %           05574
         tmp,                                                           05575
         fl,         % file number for jump name external file %       05576
         ldstr[35];          % data structure for link parsing %        05577
      LOCAL STRING                                                      05578
         jnestr[200],        % for jump name external file %            05579
         locstr[200],        % to hold dae locally %                    05580
         st1[100], st2[100], st3[100], st4[100]; %for link parsing%
                                                                        05581
      LOCAL TEXT POINTER                                                05582
         pscan, tp1, tp2, tp9, z1;                                      05583
      REF ptr, da, t1, t2, fl;                                          05584
                                                                        05585
      % initialize %                                                    05586
         savslh ← slashflg := FALSE;                                    05587
         fnlsls ← rsrng ← FALSE;                                        05588
      %save value of viewspec data%                                     05589
         vs1 ← da.davspec;                                              05590
         vs2 ← da.davspc2;                                              05591
         cacode ← da.dacacode;                                          05592
         useqgen ← da.dausqcod;                                         05593
      % cleanup on errors %                                            05594
         ON SIGNAL ELSE                                                 05595
            BEGIN                                                       05596
            slashflg ← savslh;                                          05597
```

```
        da.davspec ← vs1;                                                   05598
        da.davspc2 ← vs2;                                                   05599
        da.dacacode ← cacode;                                               05600
        da.dausqcod ← useqgen;                                              05601
        END;                                                                05602
% copy dae to minimize page faulting, etc. %                               05603
   *locstr* ← t1 t2;                                                        05604
FIND SF(*locstr*) ↑z1 ↑tp9 >;                                               05605
LOOP                                                                        05606
   BEGIN                                                                    05607
   IF inptrf THEN GOTO caeerror;                                            05608
   count ← 1;                                                               05609
   FIND z1 >;                                                               05610
   *errwrk* ← char ← READC;                                                 05611
   FIND ↑z1;                                                                05612
   CASE char OF                                                             05613
      = SP, = TAB, = EOL, = CR, = LF: REPEAT LOOP;                          05614
      = ENDCHR: NULL;                                                       05615
      ENDCASE fnlsls ← rsrng ← FALSE;                                       05616
   CASE char OF                                                             05617
      = SP, = TAB, = EOL, = CR, = LF:  NULL;                                05618
      ='.:   % positional relation %                                        05619
         LOOP                                                               05620
            BEGIN                                                           05621
            FIND z1 >;                                                      05622
            *errwrk* ← char ← READC;                                        05623
            FIND ↑z1;                                                       05624
            count ← 1;                                                      05625
            CASE char OF                                                    05626
               IN ['0, '9]:                                                 05627
                  BEGIN                                                     05628
                  FIND < CH >;                                              05629
                  count ← gadnum();                                         05630
                  *errwrk* ← char ← READC;                                  05631
                  FIND ↑z1;                                                 05632
                  CASE char OF                                              05633
                     #L: GOTO caeerror;                                     05634
                     ENDCASE REPEAT CASE 2(char);                           05635
                  END;                                                      05636
               ='B, ='b:  % back %  %SHOULD USE SEQGEN%                     05637
                  BEGIN                                                     05980
                  rsrng ← FALSE;                                            05982
                  getpr(&getbck, count, &ptr);                              05639
                  END;                                                      05981
               ='C, ='c:  % next occurrence of content %                    05640
                  BEGIN                                                     05641
                  rsrng ← FALSE;                                            05983
                  *errwrk* ← '", *conreg*, '";                              05642
                  tmp ← 1;                                                  05643
                  srchtype ← contnt;                                        05644
                  FOR tmp UP UNTIL > count DO                               05645
                     BEGIN                                                  05646
                     IF (tmp > 1) AND (conreg.L = 1) THEN BUMP              05647
                     ptr[1];                                                05648
                     specreg( $conreg, contnt, &ptr);                       05649
                     END;
```

```
    END;                                                    05650
='D, ='d:  % down %                                         05651
    BEGIN                                                   05984
    rsrng ← FALSE;                                          05985
    getpr($getsub, count, &ptr);                            05652
    END;                                                    05986
='E, ='e:  % end %                                          05653
    BEGIN                                                   05654
    rsrng ← FALSE;                                          05987
    ptr ← getvnd(ptr, da.davspec.vslev);                    05655
    ptr[1] ← 1;                                             05656
    END;                                                    05657
='F, ='f:  % file return %                                  05658
    BEGIN                                                   05659
    *errwrk* ← char ← READC;                                05660
    CASE char OF                                            05661
       ='k, ='r:  NULL;                                     05662
       ENDCASE GOTO caeerror;                               05663
    FIND ↑z1;                                               05664
    fname ← readfrring(da.dalink, count :
    rsrng);                                                 05665
    FIND SF(*[fname]*) ↑tp1;                                05666
    IF (rhstn ← lnbfls($tp1, 0, $jnestr)) #
    lhostn THEN                                             05667
       err($"Remote File Manipulation Not
       Implemented Yet");                                   05668
    ptr ← readsrring(rsrng, 0 : ptr[1],
    da.davspec, da.davspc2);                                05669
    ptr.stfile ← cloafil($jnestr);                          05670
    END;                                                    05671
='H, ='h:  % head %                                         05672
    BEGIN                                                   05988
    rsrng ← FALSE;                                          05989
    getpr($gethed, count, &ptr);                            05990
    END;                                                    05991
='L, ='l:  % indirect link %                                05674
    BEGIN                                                   05675
    rsrng ← FALSE;                                          05992
    ptr[1] ← MAX(fchtxt(getsdb(ptr)), ptr[1]);
                                                            05676
    IF (rhstn ← lnkpspc(                                    05677
       count, &ptr, $st1, $st2, $st3, $st4,
       $ldstr)) # lnostn THEN                               05678
          err($"Remote File Manipulation Not
          Implemented Yet");                                05679
    IF ldstr[fe+1] > ldstr[fs+1] THEN %filename%
                                                            05680
       ptr ← nfstid ($st2, $st3, $st4, &da :
       ptr[1], da.davspec, da.davspc2,
       da.dacacode, da.dausqcod)                            05681
    ELSE %old file%                                         05682
       IF st3.L THEN % address expression %                 05683
          BEGIN                                             05684
          FIND SF(*st3*) ↑tp1 SE(*st3*) ↑tp2;
                                                            05685
          da.davspec ← caddexp( $tp1, $tp2, &da,
```

```
                &ptr : da.davspc2, da.dacacode,
                da.dausqcod );                                  05686
                END;                                            05687
            IF st4.L # empty THEN feedlt(&da, $st4);
                                                                05688
            END;                                                05689
        ='N, ='n:  % next %  %SHOULD USE SEQGEN%                05690
            BEGIN                                               05993
            rsrng ← FALSE;                                      05994
            getpr($getnxt, count, &ptr);                        05997
            END;                                                05996
        ='O, ='o:  % origin %                                   05693
            BEGIN                                               05694
            rsrng ← FALSE;                                      05998
            ptr.stPsid ← origin;                                05695
            ptr[1] ← 1;                                         05696
            END;                                                05697
        ='P, ='p:  % predecessor %                              05698
            BEGIN                                               05999
            rsrng ← FALSE;                                      06000
            getpr($getprd, count, &ptr);                        06001
            END;                                                06002
        ='R, ='r:  % return %                                   05700
            BEGIN                                               05701
            rsrng ← FALSE;                                      06003
            readfrring(da?dalink, 0 : sring);                   05702
            ptr ← readsrring(sring, count : ptr[1],
            da.davspec, da.davspc2);                            05703
            END;                                                05704
        ='S, ='s:  % succ %                                     05705
            BEGIN                                               06004
            rsrng ← FALSE;                                      06005
            getpr($getsuc, count, &ptr);                        06006
            END;                                                06007
        ='T, ='t:  % tail %                                     05707
            BEGIN                                               06008
            rsrng ← FALSE;                                      06009
            getpr($getail, count, &ptr);                        06010
            END;                                                06011
        ='U, ='u:  % up %                                       05709
            BEGIN                                               06012
            rsrng ← FALSE;                                      06013
            getpr($getup, count, &ptr);                         06014
            END;                                                06015
        ='W, ='w:  % next occurrence of word %                  05711
            BEGIN                                               05712
            rsrng ← FALSE;                                      06016
            *errwrk* ← '", *conreg*, '", "=w";                  05713
            tmp ← 1;                                            05714
            srchtype ← wordtyp;                                 05715
            FOR tmp UP UNTIL > count DO                         05716
                BEGIN                                           05717
                IF (tmp > 1) AND (conreg.L = 1) THEN BUMP
                ptr[1];                                         05718
                specreg( $conreg, wordtyp, &ptr);               05719
                END;                                            05720
```

```
                    END;                                                05721
                 # LD:   % end of structural relation %                 05722
                    BEGIN                                               06018
                    CASE char OF                                        06022
                       = ENDCHR:  NULL;                                 06023
                       ENDCASE FIND z1 < CH ↑z1;                        06024
                    REPEAT LOOP 2;                                      06020
                    END;                                                06021
                 ENDCASE GOTO caeerror;                                 05724
              END;                                                      05725
        = '-:   % scan relation or link %                               05726
           CASE READC OF                                                05727
              = '-:     % link %                                        05728
                 BEGIN                                                  05729
                 FIND < CH >;                                           05730
                 GOTO cadlnk;                                           05731
                 END;                                                   05732
              = ENDCHR:  GOTO caeerror;                                 05733
              ENDCASE   % scan relation %                               05734
                 BEGIN                                                  05735
                 FIND < CH >;                                           05736
                 GOTO cadscr;                                           05737
                 END;                                                   05738
        ='+:   % scan relation %                                        05739
           (cadscr):                                                    05740
           BEGIN                                                        05741
           dir ← IF char = '+ THEN 1 ELSE -1;                           05742
           LOOP                                                         05743
              BEGIN                                                     05744
              FIND z1 >;                                                05745
              *errwrk* ← char ← READC;                                  05746
              FIND ↑z1;                                                 05747
              count ← dir;                                              05748
              CASE char OF                                              05749
                 IN ['0, '9]:                                           05750
                    BEGIN                                               05751
                    FIND < CH >;                                        05752
                    count ← gadnum() * dir;                             05753
                    *errwrk* ← char ← READC;                            05754
                    FIND ↑z1;                                           05755
                    CASE char OF                                        05756
                       #L: GOTO caeerror;                               05757
                       ENDCASE REPEAT CASE 2(char);                     05758
                    END;                                                05759
                 ='C, ='c: %character scan%                             05760
                    BEGIN                                               05761
                    count ← gaddir(&ptr, count);                        05762
                    UNTIL (count ← count-1) < 0 DO FIND CH;             05763
                    FIND ↑ptr;                                          05764
                    END;                                                05765
                 ='E, ='e: %statement end%                              05766
                    FIND SE(ptr) < CH ↑ptr;                             05767
                 ='F, ='f: %statement front%                            05768
                    FIND SF(ptr) ↑ptr;                                  05769
                 ='I, ='i: %invisible scan%                             05770
                    BEGIN                                               05771
```

```
                count ← gaddir(&ptr, count : scnbck);                    05772
                UNTIL (count ← count-1) < 0 DO                           05773
                   FIND $NP $PT;                                         05774
                IF scnbck THEN FIND $NP;                                 05775
                FIND ↑ptr;                                               05776
                END;                                                     05777
             ='L, ='l: %link scan%                                       05778
                BEGIN                                                    05779
                count ← gaddir(&ptr, count : scnbck);                    05780
                UNTIL (count ← count-1) < 0 DO                           05781
                   FIND CH ('( / '<);                                    05782
                FIND ↑ptr;                                               05783
                END;                                                     05784
             ='N, ='n: %number scan%                                     05785
                BEGIN                                                    05786
                count ← gaddir(&ptr, count : scnbck);                    05787
                UNTIL (count ← count-1) < 0 DO                           05788
                   BEGIN                                                 05789
                   FIND $D $(NOT D) ↑pscan;                              05790
                   % call number delimiter routine to find
                   end of number %                                       05791
                   ndr( $pscan, $tp1, $tp2 );                            05792
                   % set pointer to end of current number %
                                                                         05793
                      IF scnbck                                          05794
                         THEN FIND tp1                                   05795
                         ELSE FIND tp2 $(NOT D);                         05796
                   END;                                                  05797
                FIND ↑ptr;                                               05798
                END;                                                     05799
             ='V, ='v: %visible scan%                                    05800
                BEGIN                                                    05801
                count ← gaddir(&ptr, count : scnbck);                    05802
                UNTIL (count ← count-1) < 0 DO                           05803
                   FIND $PT $NP;                                         05804
                IF scnbck THEN FIND $PT;                                 05805
                FIND ↑ptr;                                               05806
                END;                                                     05807
             ='W, ='w: %word scan%                                       05808
                BEGIN                                                    05809
                count ← gaddir(&ptr, count : scnbck);                    05810
                UNTIL (count ← count-1) < 0 DO                           05811
                   FIND $LD $NLD;                                        05812
                IF scnbck THEN FIND $LD;                                 05813
                FIND ↑ptr;                                               05814
                END;                                                     05815
             # LD:   % end of scan relation %                            05816
                REPEAT CASE 2(char);                                     05817
          ENDCASE GOTO caeerror;                                         05818
          END;                                                           05819
       END;                                                              05820
    ='/:   % slash %                                                     05859
       BEGIN                                                             05860
       ccurcon( &ptr, $errwrk );                                         05861
       typeas( $errwrk );                                                05862
       fnlsls ← TRUE;                                                    05863
```

```
    END;                                                        05864
='\:   % backslash %                                            05865
    cprista(ptr, &da, &da);                                     05866
= '(, = '<:     %link %                                         05867
    (cadlnk):                                                   05868
    BEGIN                                                       05869
    FIND < CH ↑z1 ↑tp1;                                         05870
    lnkprs( $z1, $ldstr);                                       05871
    ldstr[lfn] ← t1.stfile; %because copied bugged link%
                                                                05872
    IF (rhstn ← lnkpspc( 1, 0, $st1, $st2, $st3, $st4,
    $ldstr)) # lhostn THEN                                      05873
       err($"Remote File Manipulations Not Implemented
       Yet");                                                   05874
    %because copied bugged link%                                05875
       IF (st3.L = empty) AND                                   05876
          t1.stfile AND (ldstr[fs+1] >= ldstr[fe+1]) THEN
                                                                05877
             *st3* ← '0, STRING(getsid(t1)), '+,
             STRING(t1[1]), 'c;                                 05878
    z1 ← ldstr[le];  z1[1] ← ldstr[le+1];                       05879
    *errwrk* ← tp1 z1;                                          05880
    IF ldstr[fe+1] > ldstr[fs+1] THEN  % file name %            05881
       ptr ← nfstid( $st2, $st3, $st4, &da, : ptr[1],
       da.davspec, da.davspc2, da.cacode, da.dausqcod)
                                                                05882
    ELSE                                                        05883
       IF st3.L THEN % address expression %                     05884
          BEGIN                                                 05885
          FIND SF(*st3*) ↑tp1 SE(*st3*) ↑tp2;                   05886
          da.davspec ← caddexp( $tp1, $tp2, &da, &ptr :
          da.davspc2, da.dacacode, da.dausqcod );               05887
          END;                                                  05888
    IF st4.L THEN feedlt( &da, $st4);                           05889
    END;                                                        05890
='": %content search%                                           05891
    BEGIN                                                       05892
    FIND < CH ↑tp1 > CH;                                        05893
    *st1* ← NULL;                                               05894
    gdlit2( $st1, '");                                          05895
    FIND ↑z1;                                                   05896
    *errwrk* ← tp1 z1;                                          05897
    (srchpar):                                                  05898
       *conreg* ← *st1*;                                        05899
       srchtype ← conspt( $z1 : count, sdomain);                05900
       FIND ↑z1;                                                05901
       *errwrk* ← tp1 z1;                                       05902
       CASE sdomain OF                                          05903
          = -1:                                                 05904
             BEGIN                                              05905
             tmp ← 1;                                           05906
             FOR tmp UP UNTIL > count DO                        05907
                BEGIN                                           05908
                IF (tmp > 1) AND (st1.L = 1) THEN BUMP
                ptr[1];                                         05909
                specreg( $st1, srchtype, &ptr);                 05910
```

```
                    END;                                                05911
                 END;                                                   05912
              ENDCASE                                                   05913
                 BEGIN                                                  05914
                 tmp ← 1;                                               05915
                 FOR tmp UP UNTIL > count DO                            05916
                    LOOP                                                05917
                       BEGIN                                            05918
                       FIND ptr ↑tpl;                                   05919
                       IF (tmp > 1) AND (stl.L = 1) THEN BUMP
                       ptr[1];                                          05920
                       specreg( $st1, srchtype, &ptr);                  05921
                       IF ptr # endfil THEN EXIT LOOP;                  05922
                       IF (sdomain ← sdomain-1) THEN                    05923
                          ptr ← getnxt(tpl)                             05924
                       ELSE EXIT LOOP 2;                                05925
                       END;                                             05926
                 END;                                                   05927
    END;                                                                05928
='' %apostrophe% : %character search%                                   05929
    BEGIN                                                               05930
    FIND < CH ↑tpl > CH;                                                05931
    *st1* ← char ← READC;                                               05932
    *errwrk* ← '', char;                                                05933
    FIND ↑z1;                                                           05934
    GOTO srcnpar;                                                       05935
    END;                                                                05936
='#: % marker %                                                         05937
    BEGIN                                                               05938
    *st1* ← NULL;                                                       05939
    gdlit2($st1, SP);                                                   05940
    FIND ↑z1;                                                           05941
    *errwrk* ← *errwrk*, *st1*;                                         05942
    ptr ← lkmkr($st1, ptr.stfile : ptr[1]);                             05943
    FIND z1 >;                                                          05944
    END;                                                                05945
= '&,    % external name %                                              05821
= '*,    % next name %                                                  05822
= '!,    % name in current branch %                                     05823
= LD, = '', = '-, = '@:    % name %                                     05824
    BEGIN                                                               05825
    tmp ← ptr.stfile;     % save file no. for jump name
    external %                                                          05826
    IF z1[1] > 2 THEN *locstr*[z1[1]-2] ← SP;                           05827
    gadname(aptr, $st2, (CASE char OF                                   05828
       = '&: extname;                                                   05829
       = '*: seqname;                                                   05830
       = '!: braname;                                                   05831
       ENDCASE nametyp));                                               05832
    FIND ↑z1;                                                           05833
    IF char = '& AND ptr = endfil THEN                                  05834
       IF getenf( tmp, $jnestr) THEN                                    05836
          BEGIN                                                         05835
          IF *jnestr* = *ojnestr* THEN                                  05837
             ptr ← jfnefln                                              05838
          ELSE                                                          05839
```

```
                        BEGIN                                                   05840
                        IF jfnefln THEN                                         05841
                           BEGIN                                                05842
                           &fl ←flntadr(jfnefln.stfile);                        05843
                           fl.flnoclos ← FALSE;                                 05844
                           END;                                                 05845
                        *ojnestr* ← *jnestr*;                                   05846
                        FIND SF(*jnestr*) ↑tp1 SE(*jnestr*) ↑tp2;
                                                                                05847
                        caddexp($tp1, $tp2, &da, &ptr);                         05848
                        jfnefln ← ptr;                                          05849
                        &fl ←flntadr(jfnefln.stfile);                           05850
                        fl.flnoclos ← TRUE;                                     05851
                        END;                                                    05852
                     IF ptr.stpsid = origin THEN *st1* ← *st2*, "
                     .1"                                                        05975
                     ELSE *st1* ← '!, *st2*, " .1";                             05976
                     FIND SF(*st1*) ↑tp1 SE(*st1*) ↑tp2;                        05854
                     caddexp($tp1, $tp2, &da, &ptr);                            05855
                     END;                                                       05856
               IF ptr = endfil THEN *errwrk* ← *st2*;                           05857
               END;                                                             05858
            = ENDCHR:                                                           05946
               BEGIN                                                            05947
               slashflg ← savsln;                                               05948
               IF (slashflg) AND (NOT fnlsls) THEN                              05949
                  BEGIN                                                         05950
                  ccurcon( &ptr, $errwrk );                                     05951
                  typeas( $errwrk );                                            05952
                  END;                                                          05953
               RETURN (da.davspec := vs1,                                       05954
                  da.davspc2 := vs2,                                            05955
                     da.dacacode := cacode,                                     05956
                        da.dausqcod := useqgen,                                 05957
                           rsrng);                                              05979
               END;                                                             05958
            ENDCASE                                                             05959
               GOTO caeerror;                                                   05960
         IF ptr = endfil THEN GOTO caeerror;                                    05961
         END;  % of loop %                                                      05962
      (caeerror): %error return%                                                05963
         slashflg ← savsln;                                                     05964
         da.davspec ← vs1;                                                      05965
         da.davspc2 ← vs2;                                                      05966
         da.dacacode ← cacode;                                                  05967
         da.dausqcod ← useqgen;                                                 05968
         SIGNAL(gaderr, $errwrk);                                               05969
      END.                                                                      05970
      %  %                                                                      05971
```

```
%.....ADDRESS EXPRESSION SUPPORT ROUTINES...%                          02313

   (conspt)          % used for content and word searches %            02969
      % determines type of search, number of elements to be found,
      and scope of the search.%                                        02970
      PROCEDURE(z1);                                                   02971
      LOCAL                                                            02972
         count, type, scope, tmp;                                      02973
      REF z1;                                                          02974
                                                                       02975
      count ← type ← scope ← tmp ← 0;                                  02976
      IF NOT FIND $(SP/TAB) '= THEN                                    02977
         RETURN( contnt, 1, -1);                                       02978
      FIND ↑z1;                                                        06029
      CASE READC OF                                                    02979
         IN ['0, '9]:                                                  02980
            BEGIN                                                      02981
            FIND < CH >;                                               02982
            tmp ← gadnum();                                            02983
            FIND ↑z1;                                                  06030
            REPEAT CASE;                                               02984
            END;                                                       02985
         = 'C, ='c:                                                    02986
            IF NOT type THEN                                           02987
               BEGIN                                                   02988
               FIND ↑z1;                                               02989
               IF tmp THEN count ← tmp := 0 ELSE count ← 1;            02990
               type ← IF scope THEN contls ELSE contnt;                02991
               IF scope THEN RETURN( type, count, scope )              02992
               ELSE REPEAT CASE;                                       02993
               END                                                     02994
            ELSE                                                       02995
               BEGIN                                                   02996
               FIND z1;                                                02997
               RETURN( type, count, IF scope THEN scope ELSE -1 );
                                                                       02998
               END;                                                    02999
         = 'W, ='w:                                                    03000
            IF NOT type THEN                                           03001
               BEGIN                                                   03002
               FIND ↑z1;                                               03003
               IF tmp THEN count ← tmp := 0 ELSE count ← 1;            03004
               type ← IF scope THEN wordls ELSE wordtyp;               03005
               IF scope THEN RETURN( type, count, scope )              03006
               ELSE REPEAT CASE;                                       03007
               END                                                     03008
            ELSE                                                       03009
               BEGIN                                                   03010
               FIND z1;                                                03011
               RETURN( type, count, IF scope THEN scope ELSE -1 );
                                                                       03012
               END;                                                    03013
         = 'S, ='s:                                                    03014
            IF NOT scope THEN                                          03015
               BEGIN                                                   03016
               FIND ↑z1;                                               03017
```

```
            IF tmp THEN scope ← tmp := 0 ELSE scope ← 1;          03018
            CASE type OF                                          03019
               = FALSE:  REPEAT CASE 2;                           03020
               = wordtyp:                                         03021
                  RETURN( wordls, count, scope );                 03022
               = contnt:                                          03023
                  RETURN( contls, count, scope );                 03024
               ENDCASE err($"NLS System Error: CONSPT");          03025
            END                                                   03026
         ELSE                                                     03027
            BEGIN                                                 03028
            FIND z1;                                              03029
            RETURN(                                               03030
               IF type THEN type ELSE contnt,                     03031
                  IF count THEN count ELSE 1,                     03032
                     scope );                                     03033
            END;                                                  03034
      ENDCASE                                                     03035
         BEGIN                                                    03036
         FIND z1;                                                 03037
         RETURN(                                                  03038
            IF type THEN type ELSE contnt,                        03039
               IF count THEN count ELSE 1,                        03040
                  IF scope THEN scope ELSE -1 );                  03041
         END;                                                     03042
END.                                                              03043
% %                                                               03044
```

```
(gadname) PROCEDURE(ptr, ast, type);                                    02341
   LOCAL TEXT POINTER tp1, tp2, tp3;                                    02342
   REF ptr, ast;                                                        02343
   FIND ↑tp1;                                                           02344
   nmdr($tp1, $tp2, $tp3);                                              02345
   *ast* ← tp2 tp3;                                                     02346
   specreg(&ast, type, &ptr);                                           02347
   FIND tp3 >;                                                          02348
   RETURN;                                                              02349
   END.

                                                                        02350
   % %                                                                  03537
```

```
(gaddir) PROCEDURE(ptr,count);                                  02351
   REF ptr;                                                     02352
   IF count < 0 THEN                                            02353
      BEGIN                                                     02354
      FIND ptr < ;                                              02355
      RETURN ( -count, TRUE );                                  02356
      END;                                                      02357
   FIND ptr > ;                                                 02358
   RETURN ( count, FALSE );                                     02359
   END.

                                                                03542
   % %                                                          03543
```

```
(getpr) PROCEDURE(proc, count, ptr);                                    02360
   LOCAL lptr;                                                          05972
   REF proc, ptr;                                                       02361
   IF count < 0 THEN %do the inverse%                                   02362
      BEGIN                                                             02363
      &proc ← CASE &proc OF                                             02364
         = $getsuc : $getprd;                                           02365
         = $getprd : $getsuc;                                           02366
         = $getsub : $getup;                                            02367
         = $getup  : $getsub;                                           02368
         = $gethed : $getail;                                           02369
         = $getail : $gethed;                                           02370
         = $getnxt : $getbck;                                           02371
         = $getbck : $getnxt;                                           02372
         ENDCASE &proc;                                                 02373
      count ← -count;                                                   02374
      END;                                                              02375
   UNTIL (count ← count-1) < 0 DO                                       02376
      BEGIN                                                             02377
      IF &proc = $getsuc AND getftl(ptr) THEN EXIT;                     02378
         % thus successor of statement with no successor is NOP %
                                                                        02379
      IF (lptr ← proc(ptr)) = endfil THEN EXIT;  %NOP%                  05973
      ptr ← lptr;                                                       02380
      ptr[1] ← 1;                                                       02381
      END;                                                              02382
   RETURN;                                                              02383
   END.

                                                                        03545
   % %                                                                  03546
```

```
(gdlit2) PROCEDURE(ast, stopchar);                                      02384
   % append characters to ast from READC until encounter a
   stopchar or the end of input.%                                       02385
   LOCAL char;                                                          02386
   REF ast;                                                             02387
   LOOP                                                                 02388
      BEGIN                                                             02389
      CASE char ← READC OF                                              02390
         =ENDCHR, =stopchar:                                            02391
            RETURN;                                                     02392
         ENDCASE                                                        02393
            *ast* ← *ast*, char;                                        02394
      END;                                                              02395
   END.

                                                                        03547
   %  %                                                                 03548
```

```
(zadnum) PROCEDURE;                                                    02397
   % extract and evaluate number from current READC source.  %
                                                                       02398
   LOCAL count, char;                                                  02399
   IF (char ← READC) NOT IN ['0, '9] THEN                              02400
      BEGIN                                                            06026
      IF char NOT= ENDCHR THEN FIND < CH >;                            06027
      RETURN(1);                                                       06025
      END;                                                             06028
   count ← char - '0;                                                  02401
   WHILE (char ← READC) IN ['0, '9] DO                                 02402
      count ← count*10 + char - '0;                                    02403
   IF char NOT= ENDCHR THEN FIND < CH >;                               02404
   RETURN(count);                                                      02405
   END.
                                                                       03550
   %  %                                                               03551
```



```
%execute link%                                                         02406
   (nfstid)  %get t-ptr to new file and address expression%            02407
      PROCEDURE(fnmstng, stnstg, vspstg, da);                          02408
      %Given two strings, the first containing a file name, the
      second an address experssion, this routine will open the
      file, and return the corresponding stid and character
      count. Branch only viewspec is added to the viewspec string
      if this is a journal message.  DA is needed for address
      expression evaluation.%                                          02409
      %-------------%                                                 02410
      LOCAL vs1, vs2, cacode, useqgen, fno, jnlfg;                     02411
      LOCAL TEXT POINTER tptr, z1, z2;                                 02412
      LOCAL STRING lkmkst[15], lkfnst[50];                             02413
      REF fnmstng, stnstg, vspstg, da;                                 02414
      %Check to see if file is a journal file; if so, save off
      number%                                                          02415
```

```
      BEGIN                                                        02417
      *lkfnst* ← 'J, z1 z2;                                        02418
      jnlfg ← TRUE;                                                02419
      END                                                          02420
   ELSE jnlfg ← FALSE;                                             02421
tptr ← origin;                                                     02422
tptr.stfile ← fno ← cloafil(&fnmstng);                             02423
   %cloafil changes fnmstng%                                       02424
%Check to see if file is a journal message file%                   02425
   IF jnlfg AND FIND SF(*fnmstng*) ('<['>]//) "JRNL" THEN
                                                                   02426
      BEGIN                                                        02427
      *stnstg* ← *lkfnst*;                                         02428
      END                                                          02429
   ELSE jnlfg ← FALSE;                                             02430
FIND SF(*stnstg*) ↑z1 SE(*stnstg*) ↑z2;                            02431
vs1 ← caddexp($z1, $z2, &da, $tptr : vs1, cacode, useqgen);
                                                                   02432
RETURN(tptr, tptr[1], vs1, vs2, cacode, useqgen);                  02433
END.
                                                                   02434
%  %                                                               03552
```

```
%......statement name lookup......%                                    02449

   (namelook)PROC(stid, astr);                                         02450
      %This routine accepts an stid and astring, an finds the
      named statemet in the file indicated by the stid, retruning
      the stid oof the statemet or endfil.  Uses non-sequential
      lookup.  Copies name astring into local string so a literal
      may be used in the call%                                         02451
      LOCAL TEXT POINTER z1;                                           02452
      LOCAL STRING namest[40];                                         02453
      REF astr;                                                        02454
      FIND SF(stid) ↑z1;                                               02455
      *namest* ← *astr*;                                               02456
      lookup($z1, $namest, nametyp);                                   02457
      RETURN(z1);                                                      02458
      END.
                                                                       02459
      %  %                                                            03554
```

```
(lookup)PROCEDURE(ptr, astrng, type);                                  02838
   %THis routine accepts a pointer, string, and type, and does
   a search through the file indicated by the pointer for the
   statemnt indicated by the string and type as follows:               02839
      type = nametyp                                                   02840
         searches the file for the first statement found with
         the indicated name.                                           02841
         Does a non-sequential search.                                 02842
         The string is modified.                                       02843
         Returns the stid of the statement as a result and in
         the pointer, or endfil on failure.                            02844
      type = nxtname                                                   02845
         Same as name, but starts from the place in the ring
         indicated by the stid in ptr.                                 02846
         Note that this also is a non-sequential search                02847
      type = seqname:                                                  02848
         Does a sequential search of the file for the next
         statemet (beginning with the one following the one
         indicated by ptr) with the indicated name.                    02849
         Returns stid in ptr, or endfil in ptr.                        02850
      type = braname                                                   02851
         Same as seqname, but search is restricted to branch
         headed by ptr.                                                02852
      type = contnt                                                    02853
         Does a sequential search of the file fo a statement
         with the content in string.                                   02854
         Search starts with the character following the one
         indicated by ptr                                              02855
         Returns same as seqname.                                      02856
      type = contls                                                    02857
         Same as content, but looks only in the current
         statement                                                     02858
      type = wordtyp                                                   02859
         Sear hes for word in string in a manner equivalent to
         content.                                                      02860
         Returns same as content.                                      02861
      type = wordls                                                    02862
         Same as wordtyp, but looks only in the current
         statement                                                     02863
      type = sid                                                       02864
         &astrng is assumed to be an sid, and te file is
         searched for a statement with a matching sid.                 02865
         Returns same as name.                                         02866
   %                                                                   02867
   %-------------------%                                              02868
   LOCAL nhash, count, sidval, p1scn, p2scn, stid;                     02869
   REF astrng, ptr;                                                    02870
   IF ptr = endfil THEN RETURN(endfil);                                02871
   CASE type OF                                                        02872
      = nametyp, =nxtname, =extname:                                   02873
         BEGIN                                                         02874
         astruc(&astrng);                                              02875
         nhash ← hash(&astrng);                                        02876
         stid ← IF type = nxtname THEN ptr ELSE 0;                     02877
         WHILE (stid ← lkupfast(ptr, stid, nhash, rnamen)) #
         endfil DO                                                     02878
```

```
                BEGIN                                                   02879
                CCPOS SF(stid);                                         02880
                xtrnam($lkupreg, $swork, -1);                           02881
                IF *lkupreg* = *astrng* THEN                            02882
                   RETURN(ptr ← stid);                                  02883
                END;                                                    02884
             RETURN(ptr ← endfil);                                      02885
             END;                                                       02886
          = seqname:                                                    02887
             BEGIN                                                      02888
             astruc(&astrng);                                           02889
             nhash ← hash(&astrng);                                     02890
             ptr[1] ← 1;                                                02891
             ptr ← getnxt(ptr);                                         02892
             END;                                                       02893
          = braname:                                                    02894
             BEGIN                                                      02895
             ptr[1] ← 1;                                                02896
             RETURN(ptr ← namingrp(ptr, ptr, &astrng, 1000));           02897
             END;                                                       02898
          = sid:                                                        02899
             RETURN(ptr ← lkupfast(ptr, 0, &astrng, rsid));             02900
          ENDCASE IF ptr[1]=empty THEN ptr[1]←1;                        02901
       UNTIL ptr = endfil DO                                            02902
          BEGIN                                                         02903
          IF inptrf THEN %have a ruboout%                               02904
             BEGIN                                                      02905
             ptr ← endfil;                                              02906
             RETURN;                                                    02907
             END;                                                       02908
          CASE type OF                                                  02909
             =seqname:                                                  02910
                IF getnam(ptr) = nhash THEN                             02911
                   BEGIN                                                02912
                   CCPOS SF(ptr);                                       02913
                   <UTILTY, xtrnam>($lkupreg, $swork, -1);              02914
                   IF <UTILTY, compas>($lkupreg, &astrng) THEN
                                                                        02915
                   RETURN;                                              02916
                   END;                                                 02917
             =contnt:                                                   02918
                IF FIND ptr > [*astrng*] ↑ptr ←ptr THEN RETURN;
                                                                        02919
             =contls: %search is limited to a single statement%
                                                                        02920
                BEGIN                                                   02921
                IF NOT FIND ptr > [*astrng*] ↑ptr ←ptr THEN ptr ←
                endfil;                                                 02922
                RETURN;                                                 02923
                END;                                                    02924
             =wordtyp:                                                  02925
                BEGIN                                                   02926
                CCPOS ptr;                                              02927
                LOOP                                                    02928
                   IF FIND [*astrng*] ↑ptr ←ptr < THEN                  02929
                      BEGIN                                             02930
```

```
                    count ← astrng.L;                                   02931
                    UNTIL (count ← count-1) < empty DO FIND CH;
                                                                        02932
                    IF FIND ↑plscn -LD AND ptr > CH -LD THEN
                    RETURN;                                             02933
                    IF NOT FIND > plscn CH THEN EXIT;                   02934
                       % this can EXIT if astrng is empty and
                                                                        02935
                       have reached end of statement %                  02936
                    END                                                 02937
                 ELSE EXIT;                                             02938
              END;                                                      02939
           =wordls:                                                     02940
              BEGIN                                                     02941
              CCPOS ptr;                                                02942
              LOOP                                                      02943
                 IF FIND (*astrng*) ↑ptr ←ptr < THEN                    02944
                    BEGIN                                               02945
                    count ← astrng.L;                                   02946
                    UNTIL (count ← count-1) < empty DO FIND CH;
                                                                        02947
                    IF FIND ↑plscn -LD AND ptr > CH -LD THEN
                    RETURN;                                             02948
                    IF NOT FIND > plscn CH THEN                         02949
                       % this can RETURN if astrng is empty and
                       have reached end of statement %                  02950
                       BEGIN                                            02951
                       ptr ← endfil;                                    02952
                       RETURN;                                          02953
                       END;                                             02954
                    END                                                 02955
                 ELSE                                                   02956
                    BEGIN                                               02957
                    ptr ← endfil;                                       02958
                    RETURN;                                             02959
                    END;                                                02960
              END;                                                      02961
           ENDCASE err(0"NLS system error");                            02962
        ptr ← getnxt(ptr);                                              02963
        ptr[1] ← 1;                                                     02964
        END;                                                            02965
     RETURN                                                             02966
     END.
                                                                        02967
     % %                                                               02968
```

```
(specreg)                                                          02570
   %Spec a register.  This procedure converts a string or a
   statement identifier to a t-pointer.  The conversion
   algorithm depends on the first character in the register,
   and on the parameter passed as the second argument,  If a
   string is being matched, the routine expects as a third
   argument the stid at which the search should begin,      02571
      If the first character is a number, and a name is being
      speced, the register is assumed to contain a statement
      number, and FECHUX is used to convert the A-string to a
      STID.  Otherwise the register is assumed to contain
      either a word or a string to be matched.             02572
      If TYPE (the second argument) is                     02573
      1, the register is assumed to contain a name;        02574
      2, the register is assumed to contain a word;        02575
      3 or 6, the register is assumed to contain a string;
                                                           02576
      4, the register is assumed to contain a statement
      identifier;                                          02577
      In all of these cases LOOKUP is called to find the
      STID.%                                               02578
   %---------------%                                       02579
   PROCEDURE(astrng, type, ptr);                           02580
   REF astrng, ptr;                                        02581
   IF astrng.L = empty                                     02582
      THEN                                                 02583
         BEGIN                                             02584
         ptr.stpsid ← origin;                              02585
         ptr[1] ← 1;                                       02586
         RETURN;                                           02587
         END;                                              02588
   IF type = nametyp OR type = seqname OR type = nxtname OR
   type = extname                                          02589
      THEN                                                 02590
         BEGIN %number, sid or name%                       02591
         ptr[1] ← 1;                                       02592
         IF *astrng*[ 1 ] IN ['0, '9]                      02593
            THEN                                           02594
               BEGIN %number or SID%                       02595
               IF *astrng*[ 1 ] = '0 AND astrng.L > 1  % sid
               given %                                     02596
                  THEN                                     02597
                     BEGIN                                 02598
                     *astrng* ← *astrng*[2 TO astrng.L];
                     %delete '0 %                          02599
                     lookup(&ptr, cvsno(&astrng), sid);
                     %lookup sid%                          02600
                     RETURN;                               02601
                     END;                                  02602
               ptr ← fechux(&astrng, ptr.stfile);          02603
               RETURN;                                     02604
               END;                                        02605
         END;                                              02606
   lookup(&ptr, &astrng, type);                            02607
   RETURN;                                                 02608
   END.
```

02609
02610

% %

```
(lkupfast)PROC(stid, start, value, field);                          02611
   LOCAL rnb, rnptr, rnt, pgindx, rngbkend, rngbkptr, fileno;
                                                                    02612
   REF rnptr, rngbkptr;                                             02613
   fileno ← stid.stfile;                                            02614
   rnt ← (rnb ← filehead[fileno]+$rngst-$filhed) + rngm;             02615
   &rnptr ← rnb+start.stblk-1;                                      02616
   IF start THEN                                                    02617
      BEGIN                                                         02618
      IF (start ← start.stwc + ring1) >= blksiz THEN  %Gone
      over to te next block%                                        02619
         BEGIN                                                      02620
         BUMP &rnptr;                                               02621
         start ← 0;                                                 02622
         END                                                        02623
      ELSE start ← start - fbhd1;                                   02624
      END;                                                          02625
   WHILE (&rnptr ← &rnptr+1) < rnt DO                               02626
      IF rnptr.rfexis THEN                                          02627
         BEGIN                                                      02628
         IF (pgindx ← rnptr.rfcore) = 0 THEN                        02629
            pgindx ← lodrfb(&rnptr-rnb+rngbas, rngtyp,
            fileno);                                                02630
         rngbkend ← (&rngbkptr ← crpgad[pgindx] + fbhd1) +
         blksiz;                                                    02631
         IF start THEN &rngbkptr ← &rngbkptr + start := 0;
         %add  in displacement if it's there%                       02632
         WHILE &rngbkptr < rngbkend DO                              02633
            IF rngbkptr.field = value THEN                          02634
               BEGIN                                                02635
               stid.stblk ← &rnptr-rnb;                             02636
               stid.stwc ← &rngbkptr-crpgad[pgindx];                02637
               RETURN(stid);                                        02638
               END                                                  02639
            ELSE &rngbkptr ← &rngbkptr + ring1;                     02640
         END;                                                       02641
   RETURN(endfil);                                                  02642
   END.
                                                                    03556
   % %                                                             03557
```

```
(namingrp) %lookup name in file starting at stid and
terminating when through with the branch headed by lbstid,
looking down levels levels.%                                    02644
   % This procedure finds the statement whose name is in
   namstr.  the search is restricted to that part of the file
   begining with STID through the branch headed by LBSTID, and
   within LEVELS levels below STID. %                           02645
   %--------------------%                                       02646
   PROC(stid, lbstid, namstr, levels);                          02647
   LOCAL rng, nhash, save;                                      02648
   LOCAL STRING locstr[100];                                    02649
   REF namstr, rng;                                             02650
   IF stid = endfil THEN RETURN(endfil);                        02651
   asbruc(&namstr);                                             02652
   nhash ← hash(&namstr);                                       02653
   save ← rubabt := FALSE;                                      02654
   LOOP                                                         02655
      BEGIN                                                     02656
      IF getnmf(stid) AND getnam(stid) = nhash THEN             02657
         BEGIN                                                  02658
         CCPOS SF(stid);                                        02659
         xtrnam($locstr, $swork, -1);                           02660
         IF *namstr* = *locstr* THEN                            02661
            BEGIN                                               02662
            rubabt ← save;                                      02663
            RETURN(stid);                                       02664
            END;                                                02665
         END;                                                   02666
      IF levels > 0 AND (stid := getsub(stid)) # stid THEN
                                                                02667
         BUMP DOWN levels                                       02668
      ELSE                                                      02669
         LOOP                                                   02670
            IF getftl(stid) THEN                                02671
               BEGIN                                            02672
               BUMP levels;                                     02673
               IF stid = lbstid OR inptrf THEN EXIT LOOP 2;
                                                                02674
               stid ← getsuc(stid); %GETSUC RETURNS THE UP IF
               TAIL%                                            02675
               END                                              02676
            ELSE                                                02677
               BEGIN                                            02678
               IF stid = lbstid OR inptrf OR stid.stpsid =
               origin THEN                                      02679
                  EXIT LOOP 2;                                  02680
               stid ← getsuc(stid);                             02681
               EXIT LOOP;                                       02682
               END;                                             02683
      END;                                                      02684
   rubabt ← save;                                               02685
   RETURN(endfil);                                              02686
   END.
                                                                02687
   % %                                                          02688
```

FINISH 0179

AUX COD

(MLK)AUXCOD (MLK)AUXCOD (MLK)AUXCOD (MLK)AUXCOD (MLK)AUXCOD (M
(MLK)AUXCOD (MLK)AUXCOD (MLK)AUXCOD (MLK)AUXCOD (MLK)AUXCOD (M

```
< NLS, AUXCOD.NLS;295, >, 16-OCT-74 13:00 DSM ;;;;

FILE auxcod % L10 to <REL-NLS>Auxcod %% (L10,) (rel-nls,auxcod.rel,) %
                                                                    02
%.....declarations.....%
                                                                    03
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4, p = 7, m = 10, s = 9;   04
   REF rawchr;                                                      05
   REF msgda, tda;                                                0870
   REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4;                         06
   SET JSYS=104B,                                                   07
      odtim = 220B, pmap=56B, haltf = 170B;                         08
%...Message Display...%                                           0677
   (dismes) %display a message to the user%
   PROCEDURE (type,astrng);                                       01102
      %This routine is called to display a message on the screen or
      tty.  The address of an A-string is usually provided in Astrng.
      Type contains an integer which determines the action taken as
      follows:                                                    01103
         Type = 0:                                                01104
            will remove any message on the screen. An A-string need not
            be given in Astrng in this case.                      01105
            The global flag MSGRESET is set TRUE.                 01106
         Type = 1:                                                01107
            the messagg will be displayed, and the routine will return
            (with the message still on the screen).               01108
            The global flag MSGRESET is set FALSE.                01109
         Type = 2:                                                01110
            causes the message to be displayed for a few seconds.
            dismes returns immediately however.                   01111
            The global flag MSGRESET is set FALSE  (TRUE when message
            is removed by msg fork pseudo-interrupt).             01112
         Type >= 1000;                                            01113
            put the message up for type milliseconds.             01114
            The global flag MSGRESET is set FALSE  (TRUE when message
            is removed by msg fork pseudo-interrupt).             01115
      %                                                           01116
      %---------------%                                          01117
      IF nldevice = offline THEN RETURN;                          01118
      IF nlmode = typewriter THEN                                 01119
         BEGIN                                                    01120
         msgreset ← TRUE;                                         01121
         IF type THEN                                             01122
            BEGIN                                                 01123
            crlf();                                               01124
            typeas(astrng);                                       01125
            END                                                   01126
         ELSE                                                     01127
            typeas($"

            ");                                                   01128
         RETURN;                                                  01129
         END;                                                     01130
      IF type = 0  AND msgreset THEN RETURN;                      01131
      IF nldevice = devlproc AND NOT tracking THEN                01132
         track();                                                 01330
```

```
      %                                                                01331
      BEGIN let fork wait some more and then retry                     01133
      stoptimer();                                                     01134
      settimer(1000, $dismes, type, astrng);                           01135
      RETURN;                                                          01136
      END;                                                             01137
      %                                                                01329
   msgreset ← TRUE;                                                    01138
   IF type = 0 THEN                                                    01139
      BEGIN                                                            01140
      IF nldevice NOT= devlproc THEN                                   01314
         typeas($"

         ");                                                           01141
      RETURN;                                                          01142
      END;                                                             01143
   %stop the timer%                                                    01144
      stoptimer();                                                     01145
   CASE type OF                                                        01146
      =1: %put up and leave%                                           01147
         dismsg(astrng);                                               01148
      =2, >=1000: %put up for a few seconds%                           01149
         BEGIN                                                         01150
         settimer(IF type >= 1000 THEN type ELSE 3000, $dismes, 0,
         0);                                                           01151
         dismsg(astrng);                                               01152
         END;                                                          01153
      ENDCASE;                                                         01154
   RETURN END.                                                         01155
                                                                       01156

(dismsg) %..display the message for DISMES..%
PROCEDURE(astrng);                                                     01157
   REF astrng;                                                         01158
   msgreset ← FALSE;                                                   01159
   crlf();                                                             01160
   typeas(&astrng);                                                    01161
   IF astrng.L < (msgda.daright-msgda.daleft)/msgda.dahinc AND NOT
   (FIND SF(*astrng*) [EOL/CR LF/) THEN                                01162
      crlf();                                                          01163
   RETURN;                                                             01164
   END.
                                                                       01165

(settimer)  %set the system timer -- after MILLISECONDS
milliseconds, PROC will be called with ARG1 thru ARG4%
PROCEDURE (milliseconds, proc, arg1, arg2, arg3, arg4);                01166
   LOCAL frkac0, frkac1, i;                                            01167
   FOR i ← 0 UP UNTIL >= 3 DO                                          01168
      IF timrset THEN !dismsg(2000)                                    01169
      ELSE EXIT LOOP;                                                  01170
   IF i >= 3 THEN stoptimer();                                         01297
   timrset ← TRUE;                                                     01171
   timrproc ← proc;                                                    01172
   timra1 ← arg1;                                                      01173
   timra2 ← arg2;                                                      01174
   timra3 ← arg3;                                                      01175
   timra4 ← arg4;                                                      01176
```

```
   frkac1 ← MIN(10000, milliseconds); %milliseconds to wait%          01177
   !sfacs(msgfrk, $frkac0); %set timer fork acs%                      01178
   !sfork(msgfrk, 120B); %start the timer fork%                       01179
   RETURN;                                                            01180
   END.                                                               01181
                                                                      01182
(stoptimer)  %stop the system timer%
PROCEDURE;                                                            01183
   %should be able to simply halt the fork, but 10X calls it an
   error to halt a fork which is already halted, STUPID STUPID%       01184
   !ffork(msgfrk); %freeze the fork%                                  01186
   !rfsts(msgfrk); %read status%                                      01187
   IF NOT r1 .A 1B1B1O THEN                                           01188
      !hfork(msgfrk); %halt the fork%                                 01189
   !rfork(msgfrk); %thaw the fork%                                    01190
   timrset ← FALSE;                                                   01191
   RETURN;                                                            01192
   END.                                                               01193
                                                                      01194
(msgpsi) %pseudo interrupt from the timer fork%
PROCEDURE;                                                            01195
   svac1 ← r1;                                                        01196
   r1 ← $svacs;                                                       01197
   !BLT r1, svacse;                                                   01198
   s ← s + 400000400B;                                                01199
   timrset ← FALSE;                                                   01200
   %call the designated procedure%                                    01201
      /timrproc/(timra1, timra2, timra3, timra4);                     01202
   r1.LH ← $svacs;                                                    01203
   r1.RH ← 0;                                                         01204
   !BLT r1, 17B;                                                      01205
   r1 ← svac1;                                                        01206
   !JSYS debrk;                                                       01207
   END.                                                               01208

(typeas) PROCEDURE (astrng); %Type a-string on tty%                    0876
   %-----------------%                                                 0877
   REF astrng;                                                        0878
   IF NOT astrng.L THEN RETURN;                                       0879
   !sout(101B, chbmty + &astrng, -astrng.L);                          0883
   RETURN;                                                            0884
   END.                                                               0885

(typech) PROCEDURE(char); %type (translated) character on tty%         01095
   LOCAL tchar, spclcharstr;                                          01096
   REF spclcharstr;                                                   01097
   %-----------------%                                                01098
   IF (r1 ← trnslo/char/) NOT= nullch THEN !pbout;                    01099
   RETURN                                                             01100
   END.                                                               01101

(crlf) PROCEDURE; %type a carriage return-line feed%                   0871
   %-----------------%                                                 0872
   !pbout(trnslo/EOL/);                                               0873
   RETURN;                                                            0874
   END.
```

```
                                                                        0875
%.....error, abort, and termination routines.....%                       010
   (goroot) PROCEDURE;  %Goto root%                                     0671
      SIGNAL(statesig, 0);                                              0672
      END.                                                              0673
   (gps) PROCEDURE;  %GOTO STATE routine%                               0674
      SIGNAL(statesig, 0);                                              0675
      END.
                                                                        0676
   (nlsrst) PROCEDURE;  %RESET NLS to rcover from castastrophic
   problems%                                                            0886
      supervisor();                                                     0887
      halt();                                                           0889
      END.
                                                                        0888
   (rerror) PROCEDURE;                                                   092
      LOCAL STRING temp[50];                                            093
      dismes(2, $"Fatal error"); %leave for 1 second%                   094
      ddgtsymbol($temp, [m .A 18M] .A 18M -1);                          095
      typeas($"Crash at PROCEDURE: ");                                  096
      typeas($ TEMP);                                                    097
      (errhlt):                                                          098
         !JSYS 147B; %Reset jsys-- cannot use symbol because of
         conflict%                                                      099
         !JSYS haltf; GOTO errhlt;                                     0100
         %In case someone is foolish enough to try to contnue%         0101
      END.                                                              0102
                                                                        0103
   (iodaterr)PROC;                                                      0104
      %Come here on an IO data Error (channel 11)%                      0105
      [levtab] ← $ioderr;                                               0106
      !JSYS debrk;                                                      0107
      (ioderr):                                                         0108
         SIGNAL(-6, $"I/O Data Error");                                 0109
         %We don't know what file the error was on, so for now just
         type a nasty mesage and signal.  deferr will rerror%           0110
      END.                                                              0111
   (thwrfp) PROCEDURE; %thaw random file pages%                         0112
      LOCAL end, ct;                                                    0113
      REF ct;                                                           0114
      end ← (&ct ← $corpst + 1) + rfpmax;                               0115
      DO IF ct.ctpnum # 0 THEN ct.ctfroz ← 0                            0116
      UNTIL (&ct ← &ct + 1) = end;                                      0117
      RETURN;                                                           0118
      END.                                                              0119
                                                                        0120
   (thwfil) PROC(filno); %thaw file pages of file%                      0121
      LOCAL end, ct;                                                    0122
      REF ct;                                                           0123
      end ← (&ct ← $corpst + 1) + rfpmax;                               0124
      DO IF ct.ctfile = filno AND ct.ctpnum # 0 THEN ct.ctfroz ← 0      0125
      UNTIL (&ct ← &ct + 1) = end;                                      0126
      RETURN;                                                           0127
      END.                                                              0128
                                                                        0129
   (pause) PROCEDURE(tim);                                              0130
```

```
   % pause for specified number of msec %                                0131
   rl ← tim;                                                             0132
   !JSYS 167B; %dismS%                                                   0133
   RETURN END.
                                                                         0134
(halt) PROCEDURE; %terminate NLS%                                        0135
   %close work station (if NLS), set continue, and issue                 0136
   terminate jsys%                                                       0137
   %---------------%                                                    0138
   IF nlmode NOT= typewriter THEN shutdis();                             0140
   ctlquit();                                                            0141
   continue ← TRUE;                                                      0142
   closeall();                                                           0143
   !JSYS haltf;                                                          0145
   GOTO rentr;   % if a person forgets and does a CONTINUE rather
   than a RENTER after an Execute Quit, then this will do the RENTER
   for him %                                                             0146
   END.                                                                  0147
                                                                         0148
(shutdis) PROCEDURE; %shut down display before leave NLS%                0149
   LOCAL STRING send[10];                                               01259
   LOCAL da, ls, end, rtend;                                             0150
   REF ls, da;                                                           0151
   CASE nldevice OF                                                      0152
      = devlproc:                                                        0153
         BEGIN                                                           0154
         lprset();                                                       0155
         cscreen();  %just to make sure%                                01327
         vspsav ← vspsav[1] ← 0; %viewspecs will be refreshed upon
         reentry%                                                        0156
         litdahandle ← ttysim ← 0;                                       0157
         clrall(0, FALSE);                                               0158
         END;                                                            0159
      ENDCASE                                                            0160
         BEGIN                                                           0161
         IF defttysim THEN                                              01209
            BEGIN                                                       01210
            vspsav ← vspsav[1] ← 0; %viewspecs will be refreshed
            upon reentry%                                               01211
            &da ← $dpyarea;                                             01212
            end ← $dpyend;                                              01213
            UNTIL &da >= end DO                                         01214
               BEGIN                                                    01215
               IF da.daexis THEN                                        01216
                  BEGIN                                                 01217
                  dealocda(&da);                                        01218
                  &ls ← da.dalsrt;                                      01219
                  rtend ← da.dalsz * lsrtl + da.dalsrt - 1;             01220
                   UNTIL &ls >= rtend DO                                01221
                     BEGIN                                              01222
                     ls.rtlsid ← 0;                                     01223
                     &ls ← &ls + lsrtl;                                 01224
                     END;                                               01225
                  END;                                                  01226
               &da ← &da + dal;                                         01227
               END;                                                     01228
```

```
            &da ← $nlsdas;                                              01229
            end ← $nlsdae;                                              01230
            UNTIL &da >= end DO                                         01231
               BEGIN                                                    01232
               IF da.daexis THEN                                        01233
                  BEGIN                                                 01234
                  dealocda(&da);                                        01235
                  &ls ← da.dalsrt;                                      01236
                  rtend ← da.dalsrt;                                    01237
                   UNTIL &ls >= rtend DO                                01238
                     BEGIN                                              01239
                     ls.rtlsid ← 0;                                     01240
                     &ls ← &ls + lsrtl;                                 01241
                     END;                                               01242
                  END;                                                  01243
               &da ← &da + dal;                                         01244
               END;                                                     01245
            litdahandle ← ttysim ← 0;                                   01246
            CASE nldevice OF                                            01247
               =imlac0, =imlac1:                                        01248
                  BEGIN                                                 01249
                  *send* ← begmsg, 2+remfudge, echda, 45B;              01250
                  !sout(dspjfn, chbmty+$send, -send.L);                 01251
                  *send* ← begmsg, 1+remfudge, remtsn;                  01252
                  !sout(dspjfn, chbmty+$send, -send.L);                 01253
                  END;                                                  01254
               ENDCASE                                                  01255
                  IF NOT SKIP !uasqd(501000001000B) THEN                01256
                     typeas($"
                     UASQD failed, shutdis");                           01257
            END;                                                        01258
         END;                                                            0215
   !sbcim(C); %turn big char input mode off%                             0216
   RETURN END.
                                                                         0217
(gofork) PROCEDURE(prcnam, injfn, outjfn, ntiw); %goto lower fork%
                                                                         0890
   %run a lower level fork with the specified process, input jfn and
   output jfn, leave ntiw psi on%                                        0891
   LOCAL                                                                 0892
      savnm,  %saved name from tenex%                                   01315
      tmode, %terminal mode word%                                        0893
      pjfn, %jfn for process%                                            0894
      pfork,  %fork handle for process%                                  0895
      tiw;  % terminal interupt word %                                   0896
   % get and remeber tenex subsystem name %                           01316
      !getnm();                                                         01317
      savnm ← r1;                                                       01318
   %create fork%                                                         0897
      r1 ← 2B11; %pass capabilites%                                      0898
      IF NOT SKIP !JSYS 152B %cfork% THEN                                0899
         err($"Can't create fork");                                      0900
      pfork ← r1;                                                        0901
   %enable capabilities%                                                 0902
      r1 ← pfork;                                                        0903
      r2 ← r3 ← -1; %enable all%                                         0904
```

```
      !JSYS 151B; %epcap%                                                    0905
   %set primary input if given jfn%                                          0906
      IF injfn # -1 OR outjfn # -1 THEN BEGIN                                0907
         r1 ← pfork;                                                         0908
         r2.LH ← injfn; r2.RH ← outjfn;                                      0909
         !JSYS 207B; %spjfn%                                                 0910
         END;                                                                0911
   %get process into the new fork%                                           0912
      IF NOT (pjfn ← lgetjfn($subdir,prcnam,$savext,gtjprf,$lit))
      THEN err($"Can't get jfn for process");                                 0913
      r1.LH ← pfork;  r1.RH ← pjfn;  !JSYS get;                              0914
   %shut down NLS display%                                                   0915
      IF nlmode = fulldisplay THEN shutdis();                                 0916
   %save terminal mode word%                                                 0917
      r1 ← 100B;  !JSYS rfmod;  tmode ← r2;                                  0918
   %reset terminal mode%                                                     0919
      r1 ← 100B;  r2 ← 205101755208;  !JSYS sfmod;                           0920
   % save tiw this fork %                                                    0921
      !rtiw( 400000B );                                                       0922
      tiw ← r2;                                                               0923
   % set tiw to ntiw for this fork %                                         0924
      !stiw( 400000B, ntiw);                                                  0925
   %start fork using entry vector%                                           0926
      r1 ← pfork;  r2 ← 0;  !JSYS 201B; %sfrkv%                              0927
   %wait for process to terminate%                                           0928
      r1 ← pfork;  !JSYS 163B; %wfork%                                       0929
   %kill fork%                                                               0930
      r1 ← pfork;  !JSYS 153B;  %kfork%                                      0931
   %release process jfn%                                                     0932
      reljfn(pjfn);                                                           0933
   % reset tiw for this fork %                                               0934
      !stiw( 400000B, tiw);                                                   0935
   %restore subsystem name%                                                  0936
      !setnm( savnm );                                                       01319
   %restore terminal mode word%                                              0939
      r1 ← 100B; r2 ← tmode; !JSYS sfmod;                                    0940
   %restore NLS display%                                                     0941
      continue ← TRUE;                                                        0942
      <INTNLS, initdis>();                                                    0943
      continue ← FALSE;                                                       0944
   %return%                                                                  0945
      RETURN END.
                                                                             0946
%...Pseudo-interrupt routines...%                                            0780
   (rubout) PROCEDURE; %rubout interupt routine%                             0781
      %---------------------%                                                0782
                                                                             0783
      %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%                     0784
                                                                             0785
      %---------------------%                                                0786
      %save registers%                                                       0787
      svacl ← r1;                                                            0788
      r1 ← $svacs;                                                           0789
      !BLT r1, svacse;                                                       0790
      IF libflg THEN killib();                                               0791
      IF rubabt:= FALSE THEN [levtab] ← $rabort                              0792
```

```
   ELSE [rubmrk] ← TRUE;                                                0793
   %clear input buffer%                                                 0794
      !cfibf(777777B); %clear input buffer%                             0795
      bufin ← buffs; % empty input buffer %                             0797
   %restore registers%                                                  0798
   !HRLZI r1, svacs;                                                    0799
   !BLT r1, 17B;                                                        0800
   r1 ← svac1;                                                          0801
   !debrk;                                                              0802
   (rabort): %come here for real aborts%                                0803
      rubsig();                                                         0804
   END.                                                                 0805

(rubsig) PROC;                                                          0806
   %simply calls signal for rubout%                                     0807
   SIGNAL(statesig);                                                    0808
   END.                                                                 0809

(stopline) PROCEDURE; % ↑S interrupt routine%                           0810
   %----------------------%                                            0811
                                                                        0812
   %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%                   0813
                                                                        0814
   %----------------------%                                            0815
   %set flag%                                                           0818
   !AOS inpstp;                                                        01320
   !JSYS debrk;                                                         0822
   END.                                                                 0823

(trapc)  PROCEDURE; %no-op control c%                                   0824
   !SOSLE ccignore;                                                     0825
   !JSYS debrk;                                                         0826
   GOTO [savchntab[2]];                                                 0827
   END.                                                                 0828

(trapo)  PROCEDURE; %no-op control o%                                   0829
   !SOSLE ccignore;                                                     0830
   !JSYS debrk;                                                         0831
   GOTO [savchntab[3]];                                                 0832
   END.                                                                 0833

(traps)  PROCEDURE; %no-op control s%                                   0834
   !SOSLE ccignore;                                                     0835
   !JSYS debrk;                                                         0836
   GOTO [savchntab[1]];                                                 0837
   END.                                                                 0838

(joctlc)PROC;                                                           0967
   %come here on a control c%                                           0968
   !JSYS 111B; %cis--clear intterupt system (can't us set because of
   duplicate symbol)%                                                   0969
   jclosfl();                                                           0970
   jctlcres();                                                          0971
   r1 ← -1; %exec%                                                      0972
   r2 ← 2B11; %↑C channel%                                              0973
   !JSYS iic; %cause interrupt%                                         0974
```

```
   r1 ← 2000; !JSYS dismS; %for TENEX timing problem%                    0975
   dismes(2,3"Journal Process Aborted by ↑C?\);                           0976
   nlsrst();                                                             0977
   END.
                                                                         0978
(jctlcres)PROC;                                                          0979
   IF NOT jsavaccess THEN disabaccess(0, jrnlaccess);                    0980
   %connect back to original directory%                                  0981
      conjdir(FALSE);                                                    0982
   %De-activate ↑C channel%                                              0983
      r1 ← 4B5;                                                          0984
      !JSYS rcm; %read channel mask--134%                                0985
      IF r1 .A 1B11 THEN                                                 0986
         BEGIN %de-activate it%                                          0987
         r1 ← 4B5;                                                       0988
         r2 ← 1B11;                                                      0989
         !JSYS dic;                                                      0990
         r1 ← 3;                                                         0991
         !JSYS dti;                                                      0992
         END;                                                            0993
   RETURN;                                                               0994
   END.                                                                  0995
                                                                         0996
(brkconnection)  PROCEDURE; %break display screen connection%            0839
   %set program counter to actual routine%                               0840
   !MOVEM r1,svac1;                                                      0841
   !MOVEI r1,brkclabel;                                                  0842
   !MOVEM r1,@levtab;                                                    0843
   !MOVE r1,svac1;                                                       0844
   !JSYS debrk;                                                          0845
   (brkclabel):                                                          0846
      rstconnection();                                                   0847
      GOTO STATE;                                                        0848
   END.
                                                                         0849
(rstconnection)  PROCEDURE; %break display screen connection%            0850
   %receive  auto-logout%                                                0851
      !atljb(-1);                                                        0852
   %break links%                                                         0853
      IF NOT SKIP !tlink(6B11 .V 777777B, 777777B) THEN NULL;            0854
      %break adviz%                                                      0855
      IF NOT SKIP !adviz(4B11) THEN NULL;                                0856
   %shut down NLS display%                                               0857
      shutdis();                                                         0858
   IF ldspjfn THEN                                                       01322
      BEGIN                                                              01323
      IF NOT SKIP !closf(ldspjfn) THEN NULL;                             01324
      reljfn(ldspjfn);                                                   01325
      END;                                                               01326
   linkcnsl ← ldspjfn ← 0;                                               0859
   %restore NLS display%                                                 0860
      %restore old format%                                               0861
         IF savnldevice NOT= nldevice THEN setdev(savnldevice);          0862
      continue ← TRUE;                                                   0863
      <INTNLS, initdis>();                                               0864
      continue ← FALSE;                                                  0865
```

```
      alldsp();                                                    0866
   chntab[1] ← savchntab[1];                                       0867
   RETURN;                                                         0868
   END.
                                                                   0869
%...call stack underflow routine...%                               0267
(uflow) PROCEDURE; %general stack underflow routine%               0268
   s ← m ← -$gstksz;                                               0269
   !HRL m,m; !HRL s,s;                                             0270
   !HRRI s,gstack; !HRRI m,gstack;                                 0271
   state ← $gstack + 2;                                            0272
   state[1] ← $supervisor;                                         0273
   state[2] ← $gstack;                                             0274
   state[3] ← $edit;                                               0275
   dismes(2, $"Call stack underflow -- report circumstances to ARC
   staff");                                                        0276
   supervisor();                                                   0277
   halt();                                                         0278
   END.
                                                                   0279
%.....help routines.....%                                          0280
(changcom)PROC(string, retparm);                                   0281
   %This procedure accepts a string as a parameter, and types out a
   message (dismes) saying tht th command has been changed, and that
   the user should consult Folklore for details%                   0282
   %Returns in the same manner as help%                            0283
   LOCAL count;                                                    0284
   LOCAL STRING msgstring[250];                                    0285
   REF string;                                                     0286
   IF (count ← string.L) + 50 >msgstring.M THEN count ← 200;       0287
   *msgstring* ← EOL, *string*[1 TO count], EOL, "Command
   Changed--see (documentation, folklore,)";                       0288
   dismes(2, $msgstring);                                          0289
   IF retparm = -1 THEN RETURN;                                    0290
   IF retparm = -2 THEN SIGNAL(statesig);                          0291
   GOTO STATE;                                                     0292
   END.                                                            0293
%.....marker code.....%                                            0294
(delmkn) PROCEDURE (fileno, name);                                 0295
   %delete the marker whose name is passed from the specified      0296
   file%                                                           0297
   %-----------------%                                             0298
   LOCAL marker, end, flhd, mkrcount, stid, aring;                 0299
   REF marker, mkrcount, aring;                                    0300
   % make sure we have a locked file %                           01309
      stid ← origin;                                              01310
      stid.stfile ← fileno;                                       01311
      lodrng( stid : &aring );                                    01312
      aring.rsub ← aring.rsub;                                    01313
   flhd ← filhdr(fileno) - $filhed;                                0301
      %subtract $filhed here instead of throughout procedure%      0302
   &marker ← flhd + $mkrtb;                                        0303
   end ← &marker + [&mkrcount ← flhd + $mkrtbl]*mkrl;              0304
   LOOP                                                            0305
      BEGIN                                                        0306
      IF &marker >= end THEN RETURN; %no such marker%              0307
```

```
      IF marker.mkname = name THEN EXIT;                                    0308
      &marker ← &marker + mkrl;                                             0309
      END;                                                                  0310
   mvbfbf(&marker+mkrl, &marker, end-&marker-mkrl);                         0311
   mkrcount ← mkrcount - 1;                                                 0312
   RETURN;                                                                  0313
   END.                                                                     0314
                                                                            0315
(delmkr) PROCEDURE (tptrs);                                                 0316
   %delete the markers in T-string specified by arg%                        0317
   LOCAL marker, end, flhd, mkrcount;                                       0318
   REF marker, mkrcount;                                                    0319
   flhd ← filndr(/tptrs/.stfile) - $filhed;                                 0320
   &marker ← flhd + $mkrtb;                                                 0321
   end ← &marker + /&mkrcount ← flhd + $mkrtbl/*mkrl;                       0322
   UNTIL &marker >= end DO                                                  0323
      IF marker.mkpsid = /tptrs/.stpsid AND                                 0324
      marker.mkccnt IN [/tptrs+1/,/tptrs+3/) THEN                           0325
         BEGIN                                                              0326
         mvbfbf( &marker+mkrl, &marker, end-&marker-mkrl);                  0327
         mkrcount ← mkrcount - 1;                                           0328
         end ← end - mkrl;                                                  0329
         END                                                                0330
      ELSE &marker ← &marker + mkrl;                                        0331
   RETURN;                                                                  0332
   END.                                                                     0333
                                                                            0334
(insmkr) PROCEDURE (bug, name);                                             0335
   %insert marker%                                                          0336
   LOCAL marker, end, flhd, mkrcount, newmkr, mkrmaxlen;                    0337
   REF marker, mkrcount, mkrmaxlen;                                         0338
   flhd ← filndr(/bug/.stfile) - $filhed;                                   0339
   &marker ← flhd + $mkrtb;                                                 0340
   &mkrmaxlen ← flhd + $mkrtxn;                                             0670
   end ← &marker + /&mkrcount ← flhd + $mkrtbl/*mkrl;                       0341
   newmkr ← TRUE;                                                           0342
   UNTIL &marker >= end DO                                                  0343
      BEGIN                                                                 0344
      IF marker.mkname = name THEN                                          0345
         BEGIN                                                              0346
         newmkr ← FALSE;                                                    0347
         EXIT;                                                              0348
         END;                                                               0349
      &marker ← &marker + mkrl;                                             0350
      END;                                                                  0351
   IF newmkr THEN                                                           0352
      BEGIN                                                                 0353
      IF mkrcount * mkrl >= mkrmaxlen THEN                                  0354
         err( "Marker table too long -- new marker not added");             0355
      BUMP mkrcount;                                                        0356
      marker.mkname ← name;                                                 0357
      END;                                                                  0358
   marker.mkpsid ← /bug/.stpsid;                                            0359
   marker.mkccnt ← /bug+1/;                                                 0360
   RETURN;                                                                  0361
   END.                                                                     0362
```

```
                                                                        0363
(seemkr) PROCEDURE (fileno, astr);                                      01260
   LOCAL marker, end=@ flhd, mkrcount, word, stid, count, char;         01261
   LOCAL TEXT POINTER tptr;                                             01262
   LOCAL STRING locstr[40], lstr2[40];                                  01263
   REF marker, mkrcount, astr;                                          01264
   flhd ← filhdr(fileno) - $filhed;                                     01265
   &marker ← flhd + $mkrtb;                                             01266
   end ← &marker + (flhd + $mkrtbl)*mkrl;                               01267
   stid ← 0;                                                            01268
   stid.stfile ← fileno;                                                01269
   *astr* ← NULL;                                                       01270
   UNTIL &marker >= end DO                                              01271
      BEGIN                                                             01272
      *locstr* ← "    ";                                                0127
      count ← 0;                                                        01274
      word ← marker.mkname;                                             01275
      UNTIL (count ← count + 1) = 6 DO                                  01276
         BEGIN                                                          01277
         char ← CASE count OF                                           01278
            =1: word.chr0;                                              01279
            =2: word.chr1;                                              01280
            =3: word.chr2;                                              01281
            =4: word.chr3;                                              01282
            ENDCASE word.chr4;                                          01283
         IF char # 0 THEN *locstr* ← *locstr*, char                     01284
         ELSE *locstr* ← *locstr*, SP;                                  01285
         END;                                                           01286
      stid.stpsid ← marker.mkpsid;                                      01287
      % get current location to astr %                                  01288
         tptr ← stid; tptr[1] ← marker.mkccnt;                          01289
         ccurloc( $tptr, $lstr2 );                                      01290
      *astr* ← *astr*, *locstr*, *lstr2*, EOL;                          01291
      &marker ← &marker + mkrl;                                         01292
      END;                                                              01293
   RETURN;                                                              01294
   END.                                                                 01295
                                                                        01296
%.....time/date routine.....%                                           0455
(getdat) %put date and time in astring passed%                          0456
   PROCEDURE(astng);                                                    0457
   dtfrmt(-1, astng); %format current date%                             0458
   RETURN;                                                              0459
   END.                                                                 0460
                                                                        0461
(dtfrmt) %put date and time in canonical form in astring passed%        0462
   PROCEDURE(tim, astng);                                               0463
   REF astng;                                                           0465
   datfrmt( tim, 20186, &astng);                                        0669
   RETURN;                                                              0473
   END.                                                                 0474
                                                                        0475
(datfrmt) %put date and time in astring passed%                         0655
   PROCEDURE(tim, frmt, astng);                                         0656
   LOCAL bytptr, count;                                                 0657
   REF astng;                                                           0658
```

```
      bytptr ← r1 ← chbptr(astng.L) + &astng;                          0659
      r2 ← tim;                                                        0660
      r3 ← frmt;                                                       0661
      !JSYS odtim;                                                     0662
      count ← slngth(bytptr, r1);                                      0663
      IF astng.L + count > astng.M THEN err(0)                         0664
      ELSE astng.L ← astng.L + count;                                  0665
      RETURN;                                                          0666
      END.                                                             0667
                                                                       0668
%.....routines to support name delimiter commands.....%                0476
   (nmdlsset) PROCEDURE  % set name delimiters for a statement %       0490
      (stid, dlleft, dlrght);                                          0491
      LOCAL rnl, sdb;                                                  0492
      LOCAL STRING str[100];                                           0493
      REF rnl, sdb;                                                    0494
      lodrng (stid : &rnl);                                            0495
      fchsdb (stid : &sdb);                                            0496
      sdb.slnmdl ← dlleft;                                             0497
      sdb.srnmdl ← dlrght;                                             0498
      CCPOS SF(stid);                                                  0499
      *str* ← NULL;                                                    0500
      xtrnam ($str, $swork, dlleft, dlrght);                           0501
      IF str.L = empty THEN                                            0502
         BEGIN                                                         0503
         sdb.sname ← 1;                                                0504
         rnl.rnamen ← 0;                                               0505
         rnl.rnamef ← FALSE;                                           0506
         END                                                           0507
      ELSE                                                             0508
         BEGIN                                                         0509
         sdb.sname ← swork1;                                           0510
         astruc($str);                                                 0511
         rnl.rnameh ← hash($str);                                      0512
         rnl.rnamef ← TRUE;                                            0513
         END;                                                          0514
      RETURN;                                                          0515
      END.                                                             0516
                                                                       0517
   (xnmdlset) PROCEDURE  % Xecute set name deilimiters for a group %
                                                                       0518
      (stid1, stid2, dlleft, dlrght, da);                              0519
      LOCAL sw, stid;                                                  0520
      REF da, sw;                                                      0521
      &sw ← openseq (stid1, stid2, da.davspec, da.davspc2, da.dausqcod,
      da.dacacode);                                                    0522
      UNTIL (stid ← seqgen(&sw)) = endfil DO                           0523
         nmdlsset (stid, dlleft, dlrght);                              0524
      closeseq (&sw);                                                  0525
      RETURN;                                                          0526
      END.                                                             0527
                                                                       0528
   (nmdlbset) PROCEDURE  % set name delimiters for a branch %          0529
      (stid1, dlleft, dlrght, da);                                     0530
      xnmdlset (stid1, stid1, dlleft, dlrght, da);                     0531
      RETURN;                                                          0532
```

```
    END.                                                                0533
                                                                        0534
(nmdlpset) PROCEDURE  % set name delimiters for a plex %                0535
    (stid1, dlleft, dlrght, da);                                        0536
    LOCAL stid2;                                                        0537
    stid1 ← plxset (stid1 :stid2);                                      0538
    xnmdlset (stid1, stid2, dlleft, dlrght, da);                        0539
    RETURN;                                                             0540
    END.                                                                0541
                                                                        0542
(nmdlgset) PROCEDURE  % set name delimiters for a group %               0543
    (stid1, stid2, dlleft, dlrght, da);                                 0544
    stid1 ← grptst (stid1, stid2 :stid2);                               0545
    xnmdlset (stid1, stid2, dlleft, dlrght, da);                        0546
    RETURN;                                                             0547
    END.                                                                0548
                                                                        0549
%.....file status routine.....%                                         0550
(fstatus) PROCEDURE(fileno, astrng, type);                              0551
    LOCAL fstfdb[25B], drn, nt, numst, fl, cnt, usd, tot, st,
    percent, i, stid;                                                   0552
    LOCAL STRING dname[25];                                             0553
    REF st, fl, astrng;                                                 0554
    &fl ← flntadr(fileno);                                              0555
    %file name%                                                         0556
        filnam( fileno, &astrng );                                      0947
        *astrng* ← *astrng*, EOL;                                       0557
    %private file?%                                                    01298
        IF rdprvsts (fileno) = $psprivate THEN                         01299
            BEGIN                                                      01300
            *astrng* ← *astrng*, "Private File";                       01301
            stid ← 0;                                                  01302
            stid.stpsid ← origin;                                      01303
            stid.stfile ← fileno;                                      01304
            IF NOT getfacc (stid, 0, 0, 0) THEN                        01305
                *astrng* ← *astrng*, " (but with no Access List)";     01306
            *astrng* ← *astrng*, EOL;                                  01307
            END;                                                       01308
    % get fdb for Modifications and Size %                            01332
        !gtfdb( fl.florig, 25B6, $fstfdb);                              0560
            %read entire FDB%                                           0561
    IF type .A 2 THEN %being modified?%                                 0558
        BEGIN                                                           0559
        drn ← fstfdb[24B].lkdirn;                                       0562
        nt ← fstfdb[24B].lkinit;                                        0563
        IF drn # 0 OR nt # 0 THEN                                       0564
            <NLS, FILMNP, lockid>(&astrng, drn, nt)                     0565
        ELSE *astrng* ← *astrng*, "File not being modified";            0566
        *astrng* ← *astrng*, EOL;                                       0567
        %browse?%                                                       0568
            IF fl.flbrws THEN *astrng* ← *astrng*, "In temporary
            modifications mode", EOL;                                   0569
        END;                                                            0570
    IF type .A 4 THEN %directory default for links%                     0571
        BEGIN                                                           0572
        <NLS, IOEXEC, gdftdir>(fileno, $dname);                         0573
```

```
    *astrng* ← *astrng*, "Default directory for links is ",          0574
       *dname*, EOL;                                                 0575
    END;                                                             0576
IF type = 7 THEN %version creation time%                             0577
    BEGIN                                                            0578
    %version 1%                                                      0579
       *astrng* ← *astrng*, "Creation date of version 1: ";          0580
       dtfrmt(fstfdb[5], &astrng);                                   0581
       *astrng* ← *astrng*, EOL;                                     0582
    %this version%                                                   0583
       *astrng* ← *astrng*, "Creation date of this version: ";       0584
       dtfrmt(fstfdb[13B], &astrng);                                 0585
       *astrng* ← *astrng*, EOL;                                     0586
    END;                                                             0587
IF type .A 1 THEN %size%                                             0588
    BEGIN                                                            0589
    %number of statements%                                           0590
       numst ← <NLS, VERIFY, crng>(TRUE, fileno);                    0591
       *astrng* ← *astrng*, STRING(numst), " statements in file",
       EOL;                                                          0592
    %structure pages%                                                0593
       &st ← filehead[fileno] + $rngst - $filhed;                    0594
       cnt ← 0;                                                      0595
       usd ← tot ← blksiz;                                           0596
       FOR i ← 0 UP UNTIL = rngm DO                                  0597
          BEGIN                                                      0598
          IF st.rfexis THEN                                          0599
             BEGIN                                                   0600
             BUMP cnt;                                               0601
             usd ← usd + st.rfused;                                  0602
             tot ← tot + blksiz;                                     0603
             END;                                                    0604
          BUMP &st;                                                  0605
          END;                                                       0606
       *astrng* ← *astrng*,                                          0607
          "Structure pages = ", STRING(cnt), '/, STRING(rngm),
          EOL;                                                       0608
    %data pages%                                                     0609
       &st ← filehead[fileno] + $dtbst - $filhed;                    0610
       cnt ← 0;                                                      0611
       FOR i ← 0 UP UNTIL = dtbm DO                                  0612
          BEGIN                                                      0613
          IF st.rfexis THEN                                          0614
             BEGIN                                                   0615
             BUMP cnt;                                               0616
             usd ← usd + st.rfused;                                  0617
             tot ← tot + blksiz;                                     0618
             END;                                                    0619
          BUMP &st;                                                  0620
          END;                                                       0621
       *astrng* ← *astrng*,                                          0622
          "Data pages = ", STRING(cnt), '/, STRING(dtbm), EOL;       0623
    %total pages%                                                    0624
       *astrng* ← *astrng*,                                          0625
          "Total pages in file = ", STRING(fstfdb[11B].RH), EOL;
                                                                     0626
```

```
        %used words%                                                    0627
           *astrng* ← *astrng*,                                         0628
              STRING(usd), " words used out of ", STRING(tot),          0629
              " words in file (=", STRING(percent ←
              (usd*100+tot/2)/tot), "%)";                               0630
           %percent used too low?%                                      0631
           IF percent < 70 AND fstfdb[11B].RH > 3 THEN                  0632
              *astrng* ← *astrng*, EOL, EOL,                            0633
                 "Try an Update File Compact to improve % used";        0634
           END;                                                         0635
        RETURN END.
                                                                        0636
%.....group allocation.....%                                            0637
    %
    This code provides for "deleteing" jobs from the group allocation
    data page for the nls commands "Execute Logout"
    %                                                                   0638
    (lockpage) PROCEDURE(core); %lock the data page%                    0639
        LOCAL                                                           0640
           count;                                                       0641
        FOR count ← 20 DOWN UNTIL = 0 DO                                0642
           BEGIN                                                        0643
           IF SKIP !AOSE @core THEN                                     0644
              BEGIN                                                     0645
              !gtad;              %get current time%                    0646
              [core+1] ← r1;      %set time of locking%                 0647
              RETURN(TRUE);                                             0648
              END;                                                      0649
           !disms (1000);         %wait a sec%                          0650
           END;                                                         0651
        RETURN(FALSE);                                                  0652
        END.                                                            0653
FINISH of AUXCOD                                                        0654
%.....query support routines.....%                                      01333
    (qport) PROCEDURE(qflg); % Query language initialization. %         01334

        % This is the entry point into the query language. %            01335
        % note that the flag qflg is not used!  Originally used for
        diferentiating between "query" and "NIC Resource Query %        01336
        % If global flag nlparse is FALSE we came from Exec directly and
        exit will be by execute quit; if nlparse is TRUE we came from nls
        and exit will do a jump file return.%
                                                                        01337
        LOCAL STRING com[100], filename[50];                            01338
        ON SIGNAL ELSE                                                  01339
           BEGIN                                                        01340
           crlf();                                                      01341
           typeas($"Error exit from query");                            01342
           IF NOT nlparse THEN %ceq()% RETURN                           01343
           ELSE                                                         01344
              BEGIN                                                     01345
              ququit();      %Jump file return%                         01346
              %GOTO STATE;%                                             01347
              RETURN;                                                   01348
              END;                                                      01349
           END;                                                         01350
```

```
   crlf();                                                          01351
   qustart($filename, $com);                                        01352
   RETURN;                                                          01353
   END.
                                                                    01354
(qustart) PROCEDURE(filename, com);  % Query parser. %              01355

   % This parser accepts a Bring command and the name of one of four
   basic files. Interrogation of a file continues with the Show
   command. Quit causes exit back to EXEC or TNLS. A question mark
   will give command language description to user.%
                                                                    01356
   LOCAL wkstid, loaded, paraml, resptr[10];                        01357
   LOCAL STRING apndir[50], tempsr[50];                             01358
   REF filename, com, paraml;                                       01359
                                                                    01360
   % Set up default directory string %                              01361
      !JSYS gjinf;                                                  01362
      gdname(r2,$tempsr);                                           01363
      *tempsr* ← '<,*tempsr*,'>;                                    01364
   *apndir* ← "<NETINFO>";                                          01365
   *filename* ← "<NETINFO>HELP1";                                   01366
   loaded ← FALSE;                                                  01367
   crlf();                                                          01368
   % load first "help" file %                                       01369
      wkstid ← quloadit( &filename, $apndir);                       01370
   LOOP % parsing loop %                                            01371
      BEGIN                                                         01372
      echon();                                                      01373
      ON SIGNAL                                                     01374
         = ofilerr:                                                 01375
            BEGIN                                                   01376
            IF NOT (FIND SF(*filename*) ['</) THEN % no directory
            name %                                                  01377
               BEGIN                                                01378
               *apndir* ← u+tempsr*;  % append user directory %     01379
               ON SIGNAL ELSE GO TO realerr;                        01380
               wkstid ← quloadit( &filename, $apndir); % Try with
               different directory name %                           01381
               END;                                                 01382
            (realerr):                                              01383
            IF sysmsg THEN                                          01384
               BEGIN                                                01385
               typeas(MESSAGE); % there WAS a dir, or failed twice %
                                                                    01386
               crlf();                                              01387
               typeas($"File not found");                           01388
               sysmsg ← 0;                                          01389
               END;                                                 01390
            *apndir* ← "<NETINFO>"; % restore default directory %
                                                                    01391
            REPEAT LOOP;                                            01392
            END;                                                    01393
         =statesig: REPEAT LOOP;                                    01394
         ELSE;                                                      01395
      crlf();                                                       01396
```

```
typeas(@"-");                                                          01397
CASE inpcuc() OF                                                       01398
   ='S: % Show command %                                               01399
      BEGIN                                                            01400
      echo(@"how ");                                                   01401
      quinlit(&com);                                                   01402
      deblank(&com);                                                   01403
      IF loaded THEN % file is present and loaded %                    01404
         ql(wkstid, &com)                                              01405
      ELSE                                                             01406
         BEGIN                                                         01407
         typeas(@"You have not specified a file yet.");                01408
         END;                                                          01409
      END;                                                             01410
   ='B: % Bring (file name) %                                          01411
      BEGIN                                                            01412
      echo(@"ring ");                                                  01413
      quinlit(&com);                 % read filename %                01414
      deblank(&com);                                                   01415
      *filename* ← *com*;                                              01416
      loaded ← TRUE;                                                   01417
      wkstid ←  quloadit( &filename, $apndir);                         01418
      END;                                                             01419
   ='D: % data base display %                                          01420
      BEGIN                                                            01421
      echo(@"ata Base of user files");                                 01422
      CASE inpcuc() OF                                                 01423
         = CA, = EOL:                                                  01424
            BEGIN                                                      01425
            *filename* ←  "<NETINFO>DATABASE";                         01426
            loaded ← TRUE;                                             01427
            wkstid ←  quloadit( &filename, $apndir);                   01428
            END;                                                       01429
         ENDCASE REPEAT LOOP;                                          01430
      END;                                                             01431
   ='R: % Resource notebook display %                                  01432
      BEGIN                                                            01433
      echo(@"esource Notebook");                                       01434
      CASE inpcuc() OF                                                 01435
         = CA, = EOL:                                                  01436
            BEGIN                                                      01437
            *filename* ←  "<NE,INFO>RESOURCES";                        01438
            *apndir* ← "<NETINFO>";                                    01439
            loaded ← TRUE;                                             01440
            wkstid ← quloadit( &filename, $apndir );                   01441
            END;                                                       01442
         ENDCASE REPEAT LOOP;                                          01443
      END;                                                             01444
   ='A: % arpanet news display %                                       01445
      BEGIN                                                            01446
      echo(@"RPANET NEWS");                                            01447
      CASE inpcuc() OF                                                 01448
         = CA, = EOL:                                                  01449
            BEGIN                                                      01450
            *filename* ←  "<HELP>ARPANEWS";                            01451
            *apndir* ← "<NETINFO>";                                    01452
```

```
                loaded ← TRUE;                                          01453
                wkstid ← quloadit( &filename, $apndir );                01454
                END;                                                    01455
             ENDCASE REPEAT LOOP;                                       01456
          END;                                                          01457
       ='I: % Ident file display %                                      01458
          BEGIN                                                         01459
          echo($"dent File");                                           01460
          CASE inpcuc() OF                                              01461
             = CA, = EOL:                                               01462
                BEGIN                                                   01463
                *filename* ←  "<IDENTFILE>IDENTS.MASTER";               01464
                loaded ← TRUE;                                          01465
                wkstid ← quloadit( &filename, $apndir );                01466
                END;                                                    01467
             ENDCASE REPEAT LOOP;                                       01468
          END;                                                          01469
       ='N: % Start over -- NIC command %                               01470
          BEGIN                                                         01471
          echo($"ic");                                                  01472
          CASE inpcuc() OF                                              01473
             = CA, = EOL:                                               01474
                BEGIN                                                   01475
                *filename* ←  "<NETINFO>HELP1";                         01476
                loaded ← FALSE;                                         01477
                wkstid ←  quloadit( &filename, $apndir);                01478
                END;                                                    01479
             ENDCASE REPEAT LOOP;                                       01480
          END;                                                          01481
       ='?: % help, then back to previous file if any %                 01482
          BEGIN                                                         01483
          CASE inpcuc() OF                                              01484
             = CA, = EOL:                                               01485
                BEGIN                                                   01486
                crlf();                                                 01487
                wkstid ← quloadit( $"<NETINFO>HELP", $apndir );
                                                                        01488
                IF loaded THEN                                          01489
                   BEGIN                                                01490
                   typeas($" Please wait ");                            01491
                   wkstid ← quloadit( &filename, $apndir );             01492
                   END;                                                 01493
                END;                                                    01494
             ENDCASE REPEAT LOOP;                                       01495
          END;                                                          01496
       ='H: % help, then back to previous file if any %                 01497
          BEGIN                                                         01498
          echo($"elp");                                                 01499
          CASE inpcuc() OF                                              01500
             = CA, = EOL:                                               01501
                BEGIN                                                   01502
                crlf();                                                 01503
                wkstid ← quloadit( $"<NETINFO>HELP", $apndir );
                                                                        01504
                IF loaded THEN                                          01505
                   BEGIN                                                01506
```

```
                    typeas($" Please wait ");                          01507
                    wkstid ← quloadit( &filename, $apndir );           01508
                    END;                                               01509
                 END;                                                  01510
              ENDCASE REPEAT LOOP;                                     01511
           END;                                                        01512
        %='V:  viewspecs  not needed (show and bring both force
        viewspecs)%                                                    01513
           %BEGIN%                                                     01514
           %echo($"iewspecs: Type ");%                                 01515
           %getvsp(); gets string from keyboard%                       01516
           %tresiev(tda.dacsp); take care of relative levels%          01517
           %&param1 ← xviewspecs($resptr,1);                           01518
           cspupdate ← lda();                                          01519
           cspvs ← param1;                                             01520
           cspvs[1] ← param1[1] ;                                      01521
           dpset(dspyes,(cspupdate).dacsp,endfil,endfil);              01522
           cmdfinish();                                                01523
           END;%                                                       01524
        ='Q: % quit %                                                  01525
           BEGIN                                                       01526
           echo($"uit ");                                              01527
           CASE inpcuc() OF                                            01528
              = CA, = EOL:                                             01529
                 BEGIN                                                 01530
                 IF NOT nlparse THEN %ceq()% RETURN                    01531
                 ELSE                                                  01532
                    BEGIN                                              01533
                    %Jump file return%                                 01534
                    ququit();                                          01535
                    %GOTO oldgps;%                                     01536
                    RETURN;                                            01537
                    END;                                               01538
                 END;                                                  01539
              ENDCASE REPEAT LOOP;                                     01540
           END;                                                        01541
        ENDCASE                                                        01542
           BEGIN                                                       01543
           crlf();                                                     01544
           typeas($"Not recognized");                                  01545
           END;                                                        01546
     END;                                                              01547
  END.
                                                                       01548
(quloadit) PROCEDURE( filename, apndir );                              01549
  LOCAL fileno, wkstid;                                                01550
  LOCAL STRING qsname[100];                                            01551
  REF filename, apndir;                                                01552
  *qsname* ← *filename*;                                               01553
  IF NOT (FIND SF(*qsname*) ['<]) THEN % filename has no directory
  %                                                                    01554
     *qsname* ← *apndir*, *filename*;                                  01555
  %xlf($qsname, &tda);                                                 01556
  sysmsg ← 0;                                                          01557
  frefint();                                                           01558
  fileno ← tda.dacsp.stfile;%                                          01559
```

```
   % --- new code ---%                                                    01560
      fileno ← cloafil(sqsname);                                          01561
      curmkr ← orgstid;                                                   01562
      curmkr.stfile ← fileno;                                             01563
      curmkr[1] ← 1;                                                      01564
   *apndir* ← "<NETINFO>";                                                01565
   crlf();                                                                01566
   typeas($"--------");                                                   01567
   feedlt(&tda,$"BW");                                                    01568
   wkstid ← origin;                                                       01569
   wkstid.stfile ← fileno;                                                01570
   IF (wkstid ← getsub(wkstid)).stpsid = origin THEN                      01571
      BEGIN                                                               01572
      typeas($"File is empty");                                           01573
      END                                                                 01574
   ELSE                                                                   01575
      BEGIN                                                               01576
      feedlt(&tda, $"esb");                                               01577
      printg(&tda,wkstid, wkstid, brnchv,0);                              01578
      END;                                                                01579
   crlf();                                                                01580
   typeas($"--------");                                                   01581
   RETURN( wkstid );                                                      01582
   END.                                                                   01583
(q1) PROCEDURE (orstid, com); %converts "show" to appropriate jump%       01584

   % given the string typed by the user, find a statement by that
   name in the current file.%                                             01585
   LOCAL prvstid, wkstid;                                                 01586
   LOCAL TEXT POINTER z1, z2, z3, z4;                                     01587
   LOCAL STRING com1[100], erout[100];                                    01588
   REF com;                                                               01589
   %Initialize stids %                                                    01590
      wkstid ← prvstid ← orstid;                                          01591
   feedlt(&tda, $"BW"); % Don't indent; all lines, all levels %           01592
   *erout* ← NULL;                                                        01593
   IF NOT (FIND SF(*com*) ↑z1 [':] ↑z2 ←z2 ↑z3 [ENDCHR] ↑z4) THEN

      % User has typed a command like: show xyz CR %                     01594
      BEGIN                                                               01595
      wkstid ← namingrp(prvstid, prvstid, &com, 1000);                    01596
      IF wkstid # endfil THEN                                             01597
         BEGIN                                                            01598
         quprout( wkstid );                                               01599
         RETURN;                                                          01600
         END                                                              01601
      ELSE                                                                01602
         BEGIN                                                            01603
         wkstid ← namingrp(orstid, endfil, &com, 1000);                   01604
         IF wkstid # endfil THEN                                          01605
            BEGIN                                                         01606
            quprout( wkstid );                                            01607
```

```
            RETURN;                                                 01608
            END;                                                    01609
         crlf();                                                    01610
         *erout* ← *com*," not found.";                             01611
         typeas($erout);                                            01612
         RETURN;                                                    01613
         END;                                                       01614
      END                                                           01615
   ELSE                                                             01616
      % User has typed: show xyz:abc CR
      which means: find a branch named abc anywhere within the
      branch named xyz. %                                           01617
      BEGIN                                                         01618
      *com1* ← z3 z4;                                               01619
      *com* ← z1 z2;                                                01620
      wkstid ← namingrp(orstid, endfil, &com, 1000);                01621
      IF wkstid = endfil THEN                                       01622
         BEGIN                                                      01623
         crlf();                                                    01624
         *erout* ← *com*," not found.";                             01625
         typeas($erout);                                            01626
         RETURN;                                                    01627
         END;                                                       01628
      prvstid ← wkstid;                                             01629
      wkstid ← namingrp(prvstid, prvstid, $com1,1000);              01630
      IF wkstid # endfil THEN                                       01631
         BEGIN                                                      01632
         quprout( wkstid );                                         01633
         RETURN;                                                    01634
         END                                                        01635
      ELSE                                                          01636
         BEGIN                                                      01637
         crlf();                                                    01638
         *erout* ← *com1*, " not found under ", *com*, '.;          01639
         typeas($erout);                                            01640
         RETURN;                                                    01641
         END;                                                       01642
      END;                                                          01643
   END.
                                                                    01644
(quprout) PROCEDURE (stid);                                         01645
   feedlt(&tda, $"esb");                                            01646
   qubing(stid);                                                    01647
   printg(&tda,stid, stid, brnchv,0);                               01648
   RETURN;                                                          01649
   END.
                                                                    01650
(qubing) PROCEDURE (stid); % process embedded viewspecs.%           01651

   % When a statement name is followed by a string such as (uv:)
   this procedure recognizes uV as viewspecs and turns them on.%
                                                                    01652
   LOCAL STRING vstrng[10];                                         01653
   LOCAL char;                                                      01654
   *vstrng* ← NULL;                                                 01655
```

```
    s2work ← stid;                                                      01656
    s2work[1] ← fchtxt(getsdb(stid));                                   01657
    fechcl(forward,$s2work);                                            01658
    IF (char ← READC($s2work)) # '( THEN RETURN;                        01659
    IF (char ← READC($s2work)) # ': THEN RETURN;                        01660
    LOOP                                                                01661
       BEGIN                                                            01662
       char ← READC($s2work);                                           01663
       IF char # ') THEN *vstrng* ← *vstrng*,char                       01664
       ELSE                                                             01665
          BEGIN                                                         01666
          feedit(&tda, $vstrng);                                        01667
          RETURN;                                                       01668
          END;                                                          01669
       END;                                                             01670
    END.                                                                01671

(ququit) PROCEDURE; % Quit and restore nls context. %                   01672

    % This procedure is called by the parser when the quit command is
    encountered and entry was from nls.%                                01673

    LOCAL STRING str[100];                                              01674
    %gadjf($bl, $propls, $str, 1, &tda);%                               01675
    %mvcbl();%                                                          01676
    %dismes(2,$"Ququit Called");%                                       01677
    RETURN;                                                             01678
    END.                                                                01679

(quinlit) PROCEDURE(com); %read a string, handle special
characters.%                                                            01680

    % Uses inpcuc iteratively until it finds CA or EOL (in which case
    it returns TRUE). Handles CD and BC normally. Question mark
    forces the string 'H into com. This can be used as third-level
    help but no current file takes advantage of it. Feature could be
    taken out without effect on other procedures.%                      01681

    LOCAL char;                                                         01682
    REF com;                                                            01683
    LOOP                                                                01684
       BEGIN                                                            01685
       *com* ← NULL;                                                    01686
       LOOP                                                             01687
          BEGIN                                                         01688
          char ← inpcuc();                                              01689
          CASE char OF                                                  01690
             =BC: IF com.L > empty THEN                                 01691
                BEGIN                                                   01692
                todco(*com*[com.L]);                                    01693
                bkc(&com);                                              01694
                END;                                                    01695
             =CD: SIGNAL(statesig, 0);                                  01696
             =CA: RETURN(TRUE);                                         01697
             =EOL: RETURN(TRUE);                                        01698
             ='?:                                                       01699
                                                                        01700
```

```
                BEGIN                                                   01701
                *com* ←'H ;                                             01702
                RETURN(TRUE);                                           01703
                END;                                                    01704
              ENDCASE *com* ← *com*, char;                              01705
            END;                                                        01706
        END;                                                            01707
    END.
                                                                        01708
(deblank) PROCEDURE (string);  %deblank the string%                     01709

    % Eliminate leading blanks for show and bring command. %
                                                                        01710
    LOCAL TEXT POINTER z1, z2;                                          01711
    REF string;                                                         01712
    IF FIND SF(*string*) ↑z1 $NP ↑z2 THEN                               01713
        ST z1 z2 ← NULL;                                                01714
    RETURN;                                                             01715
    END.                                                                01716
```

BDATA.

BDATA BDATA BDATA BDATA BDATA BDATA BDATA BDATA BDATA
BDATA BDATA BDATA BDATA BDATA BDATA BDATA BDATA BDATA

```
< NLS, BDATA.NLS;10, >, 19-SEP-74 21:22 HGL ;;;;
FILE bdata   %  L10   <REL-NLS>BDATA   %% (L10,) (rel-nls,BDATA.rel,) %
                                                                    01080
REGISTER %here so DDT will use these on printout%                     03
   r1=1, r2=2, r3=3, r4=4, r5=5, p=7, wa=8, s=9, m=10, rp=11,         04
   a1=12, a2=13, a3=14, a4=15;                                        05
% stack sizes %                                                      06
   SET EXTERNAL                                                     07
      %call and multiple result stack lengths%                    0963
         groupx =  4,     %maximum group number%                  0964
      %Number of pages to be mapped out for processors%            010
         freesz = 154B,                                            011
      %substitute%                                                 012
         subnum = 30,                                              013
         subnm1 = 120, %=1card*subnum -- see (UTILTY, card)%       014
      % for stack of user program addresses %                      015
         upgsksz = 20;                                             016
% *** GLOBAL DATA FOR PAGE 0 *** %                                0852
% Special stuff declared in page 0, loaded at 140B %              0859
   %...KEEPTHESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%    01082
   DECLARE EXTERNAL %...important stuff...%                       0861
   %...file block IO...%                                          0865
      rfpmin = 5,       % Index of lowest file page %             0866
      nxpg,            %index into corpst for next page%         0867
      corpst[100B],    %core page status table%                  0868
      filehead[26],    %array of filhdr(fileno) values%          0869
      filepart[26],    %array of -- TRUE if partial copy for file% 0870
   %...often used work areas...%                                  0871
      swork, % string reading work area %                         0872
         swork1[6],                                               0873
      s2work[7], % second string reading work area %              0874
      stcwrk, % string construction work area %                   0875
         stcwr1[6],                                               0876
   %...seqgen flags...%                                           0877
      inptrf,  %rubout and ↑O flag%                               0878
      rubmrk = inptrf,                                            0879
      inpstp, % ↑S flag %                                         0880
      rubabt,          %true if rubout causes abort%              0881
   %...editing global variables...%                               0882
      %...for new ring elements or data blocks...%                0883
         dblst,   % index of last used data file block %          0884
         rnglst,  % index of last used ring file block %          0885
   %...set command and aptstr routine...%                         0896
      mkrstay,         %true if do not move markers%              0897
      modeoff=177B;  %  option set  %                            0903
% sequence generator declarations %                               024
   DECLARE EXTERNAL                                               025
   sqsavsw,   % global to save argument while switching stacks %  026
   sqgwas [36],       % 4 sequence generator work areas %        027
   sqgaend,                                                       028
   sqsvws [260];       % 4 statement vector work areas; each is svmxlev +
   1 words long %                                                 029
                                                                  030
% TEMPS AVAILABLE TO ANY COMMAND %                              01055
   DECLARE EXTERNAL cm1tmp, cm2mp, cm3tmp, cm4tmp, cm5tmp, cm6tmp; 01056
% jump name external file stid and name string %                01010
```

```
   DECLARE EXTERNAL                                                    01011
      jfnefln = 0;                                                     01012
   DECLARE EXTERNAL STRING                                             01014
      ojnestr[200];                                                    01015
DECLARE EXTERNAL                                                        0126
   msstcount=0,                                                         0128
   mstxcount=0,                                                         0129
   msflag=0;  %indicates whether use measurements being recorded%       01083
% display core block requirements     %                                0133
% Declarations for NLS submodes%                                       0136
   % NLS utility subsystem %                                           01074
      DECLARE EXTERNAL                                                 01075
         tskerrcnt;     % cnt of errors for run tasks %                01076
   % user program stuff %                                              0137
      DECLARE EXTERNAL                                                 0138
        upgbuf = 554000B,    % start of user program buffer %          0139
        upgffbuf = 554000B,  % first free cell in upgbuf %             0143
        upgbend = 561777B,  % current end of user program buffer %     0140
        upgbsz = upgbdf,  % current size of user program buffer (in
        pages) %                                                       0141
        upgcacnt,  % number of Content analyzer patterns compiled %
                                                                       0142
        upgskix,   % index into upgstk %                              0144
        upgstk[upgsksz];   % stack of addresses of user programs %    0145
        DECLARE EXTERNAL STRING                                        0146
           upgnms[200];   % string to hold names of user programs %
                                                                       0147
   % run tenex subsystem stuff %                                       0990
      DECLARE EXTERNAL                                                 0991
         tiw = 0,       % saved terminal interrupt word for this fork %
                                                                       0992
         tmode   0,     % saved terminal mode word %                   0993
         trmcod = 0,    % termination char terminal code %             0994
         ojfn = 0,      % -1 or output file jfn %                      0995
         ijfn = 0,      % -1 or input file jfn %                       0996
         pjfn = 0,      % jfn for the program %                        0997
         infork = 0;    % fork handle of inferior tenex subsystem %    0998
   % CALCULATOR strings %                                              0175
      DECLARE EXTERNAL STRING                                          0176
         opstring[30],   %char represetation, current operand%         0177
         acstring[30],   %char representation, current accum%          0178
         signstr[5],    %number of accum used as operand%             0179
         tstrng[5],     %number ofaccum used as operand%              0180
         astrng[5];     %number of the accumulator being used%        0181
   %Journal%                                                           0182
      DECLARE EXTERNAL                                                 0183
         interflag, %interrogate mode flag%                            0184
         tmplt, %template submission flag%                             0185
         sabtfg, %submit abort flag%                                   0186
         jworkstid, %stid of jwork file%                               0187
         jcatstid, %stid of jcat%                                      0188
         jrnlstid, %Journal message file%                              0189
         diststid, %distributionn file%                                0190
         ctlstid, %stid for on-line distribution control files%        0191
         jdebug = 1, %true for debugginh...files go under duvall%      0192
         rfcflg, %True if RFC number has been assignnd%                0193
```

```
    jdirn, %for saving number of connected directory (journal)%
                                                                    0194
    idirn, %for saving number of connected directory (ident)%       0195
    nsdcktime,  %next (gtad) time to check if system going down%
                                                                    01030
    lavtabad, %getab table address for load average%                01031
    jdno, % address of string with list of numbers to be delivered;
    zero if none %                                                  01052
    jdid, % address of string with list of idents whose mail is to
    be processed-- zero if none %                                   01053
    jdfi, % address of string with list of files to be processed--
    zero if none %                                                  01054
    idfcnt, %counter for opening and closing ident file%            01021
    idfno, %file number of ident-file or zero%                      01022
    ckiWheel=1, %true if should check for identWheels when open
    identfile%                                                      01024
    idcrec, %address of current record string%                      01023
    idcrtype, %current record type (indtyp, grptyp, or orgtyp)%
                                                                    01027
    idcident, %address of current record ident string%              01025
    nwrecflag = 0, %true if user is defining a new record for the
    ident-file%                                                     01026
    idmodflag, %true if user is allowed to modify the current
    record%                                                         01048
    idmodified, %true if user has modified the current ident
    record%                                                         01049
    oljmlav=2026B8, %max load average (in floating point) for
    running on line del%                                            0196
    oljsbn, %six bit journal system name%                           0197
    %Now timing variables%                                          0198
       jtimfg, %True if timing%                                     0199
       jt0, jt1, jt2, jt3, jt4, jt5, jt6, jt7, jt8, jt9, jt10,
       jt11, jt12, jt13, jt14, jt15, %for individual times..see
       jtime for meanings%                                          0200
    ojsqsw, %seq work area for net jounal delivery%                 0201
    jsavaccess; %for restoring file access after Journal has run%
                                                                    0202
DECLARE EXTERNAL TEXT POINTER                                       0203
    jwpl, %pointer used to locate header in jwork%                  0204
    tmpbug, %pointer to user journal template%                      0205
    jb1, jb2; %misc pointers needed by Journal%                     0206
DECLARE EXTERNAL STRING                                             0207
    ipassw[40], %for saving password of connected director
    (ident)y%                                                       0208
    jpassw[40], %for saving password of connected directory
    (journal)%                                                      0209
    rfcnum[5], %for saving off RFC pumber if one is assigned%       0210
    datesr[20], %for date/time of entry%                            0211
    jnamstr[50]; %for building file names used by Journal%          0212
%Identification system%                                             0214
DECLARE EXTERNAL STRING                                             0215
    identstr[200], %used for building new entries%                  0216
    idwork[50], %work area%                                         0217
    newid[50], %used in collection of new id's%                     0218
    idntdel = "?
    .()/\+!#$%←=@;*:<>[] ",                                         01086
```

```
      cmntdl = ")", %terminators for comments in identlists%      0221
      idnamdel[5];  %contains delimiters for reading names %      0222
   DECLARE EXTERNAL STACK jidstk[20, 2];                          0223
   DECLARE EXTERNAL identwheel;  %true if user is a priviledged Ident
   system user%                                                   0224
   DECLARE EXTERNAL jidsbot;                                      0225
%Number system%                                                   0226
   DECLARE EXTERNAL
      numstid; %stid of number file%                              0228
%Hard copy Journal distribution%                                  0229
   DECLARE EXTERNAL                                               0230
      % hcdiststid, Sometimes used for Global stid into hcdistfile%
                                                                  0231
      lptjfn, % Line ptr JFN--used globally by printing machinery%
                                                                  0232
      lptused, %Used as a count of number of documents which have
      been printed since the line printer was opened%             0233
      docjfn, %JFN of Document portion of document being printed%
                                                                  0234
      hdrjfn, %JFN of header portion of document being printed%   0235
      docstrt, %Character displacement into docjfn file to bypass
      header%                                                     0236
      pfilnum, %file number used to tell printing machiinery when to
      change the document file%                                   0237
      lastfnum, %Used in conjunction winth lastfnum%              0238
      prntfg, %True if anything has been priinted this running of
      hard copy distribution%                                     0239
      mastacprintflag, %print master and access copies on this
      machine%                                                   01029
      idfile; %Occaisionally used globally for file number of ident
      file%                                                       0240
   DECLARE EXTERNAL STRING                                        0241
      oprocn[50], %Contains name of Output Processor%             0242
      ptstr[50], %Name of printer file--LPT or other%             0243
      opstr[50], %Contains ident of operator%                     0244
      docwfn = "<journal>DPNTWRK.txt;0
      ", %Name of document print work file%                       0245
      hdrwfn = "<journal>HPNTWRK.txt;0
      "; %Name of header print work file%                         0246
%FTP routines%                                                    0247
   DECLARE EXTERNAL ftphnd = 0, ftpem, ftperm, ftpoh, ftpoha;     0843
%collector sorter%                                                0250
   DECLARE EXTERNAL                                               0251
      keypr,                                                      0252
      vcvulp,                                                     0253
      vcvtp,                                                      0254
      mersiz,                                                     0255
      merrff,                                                     0256
      vtop, %vector index%                                        0257
      infopn, %flag--true if input file open%                     0258
      cstid, %stid of output file%                                0259
      outcnt, %count of statements in current output file%        0260
      smtmax = 10000, %maximum numbr of statements in one output
      file%                                                       0261
      colda, %address of display area%                            0262
      colsw, %addess of sequence area%                            0263
```

```
        sortfg, %true if we are to sort%                                    0264
        lngflg, %true if length is considered befor value in sort%          0265
        namndx, %index into input file name list%                           0266
        namlst[50], %pointers to text for input file names (indexed by
        naamndx%                                                            0267
        nambuf[100], %text area for input file names%                       0268
        ninfil, %number of input files%                                     0269
        strpkey, %true if delete keys%                                      0270
        cversion; %Number in sequence of output file%                       0271
      DECLARE EXTERNAL STRING outnam[50]; %contains root name for output
      files%                                                                0272
    %Catalog system%                                                        0273
      DECLARE EXTERNAL                                                      0274
        cppflag = FALSE; % Catalog Production Processor in Operation
        Flag %                                                              0275
% DECLARE's %                                                             0276
    DECLARE EXTERNAL                                                        0277
      exp=0,      %experimental system flag%                                0279
      hostni = -1, % LH is HOSTN table #; RH is # entries in HOSTN %
                                                                            0818
      hostsi = -1,  % LH is HSTNAM table #; RH is # entries in HSTNAM %
                                                                            0819
      ttytbl,    % system JOBTTY table number %                           01000
      jobtbl,    % system JOBDIR table number %                           01001
      bndchk,  % TRUE => perform boundary checks at tt (qt) %             01079
      oprtty,     %operator tty (gets journal errors, archive ret
      requests%                                                             0999
      lstred,  % last read date of initial file for user programs %
                                                                            01070
    %....CALCULATOR.....%                                                   0309
      %accumulators %                                                       0310
        accum[20],                                                          0311
        asub,      %subscript of current accumulator%                       0312
        vaccum[2],  %scratch accum for EVALUATE%                            0313
        opfloat[2], %double floating current operand%                       0314
        cbadent = 0, %TRUE if user enters calc from the calc %              0823
        nofeedbk,    % true if TNLS user set terse feedback %              0315
      % misc %                                                              0316
        cda,       %display area address, may be actual or pseudo%         0317
        castid,    %current location in CALC-IDENT file%                    0318
        proc,     %addr of arithmetic execution routine%                    0319
      % format variables %                                                  0320
        cadflg = 0,                                                         0321
        calflg = 0,                                                         0322
        cacflg = 0,                                                         0323
        dfoutm = 064014120200B; % mask for dfout JSYS %                     0324
    DECLARE EXTERNAL STRING                                                 0344
      vrsnno[5],  %save area for version number%                            0352
      ladj[40],   %string forlevel adjust%                                  0360
      sfhead[80],                                                           0361
      sfstng[100],                                                          0362
      lkupreg[40], %<JUMP, lookup> string save area%                        0370
      prbuf[120];  %typewriter output buffer%                               0371
    DECLARE EXTERNAL TEXT POINTER                                           0396
      p3,  p4,  p5,  p6,  p7,  p8,  p9,  p10;                               0397
    DECLARE EXTERNAL                                                        0400
```

```
%..?file status table...%                                                0406
   filst[100], %1 entry per file, 25 (filmax) files,  4 (filstl)
   words per entry%                                                      0407
   filcnt = 0,                                                           0408
%...substitute work area...%                                            0495
   subrec[subnml],                                                       0496
   subhed[4], %=lshed -- see (UTILTY, shed)%                             0497
%...file access stuff...%                                               0501
   access = 1, %normal access%                                          0502
   accmask = (1, 2, 4, 10B, 20B, 40B, 100B, 200B),                      0503
   normlaccess = 0,                                                      0504
   jrnlaccess = 1,                                                       0505
   debugaccess = 1,                                                      0506
%...file header...%                                                     0507
   % DONT CHANGE THE ITEMS IN THE HEADER %                              0508
   filhed[5],                                                            0509
      % these extra words may be taken for additions to header%         0510
   fcredt,    % file creation date %                                     0511
   nlsvwd = 1,    % nls version word (keyword) %                        0512
   sidcnt,    %count for generating SID's%                              0513
   finit,     % initials at last write %                                 0514
   funo,      % user number (file owner) %                               0515
      %if <0, RH is pointer to string in fileheader%                    0844
   lwtim,     % last write time %                                        0516
   namdl1,    % left name delimiter default character %                  0517
   namdl2,    % right name delimiter default character %                 0518
   rngl,      % upper bound on data ring file blocks %                   0519
   dtbl,      % upper bound on data file blocks %                        0520
   rfbs[6],     % start of random file block status tables %             0521
   rngst[95],  % ring block status table %                              0522
   dtbst[370], % data block status table %                              0523
   mkrtxn = 20,    % marker table maximum length %                       0524
   mkrtbl,      % number of markers in marker table %                   0525
      %each marker takes MKRL words%                                    0845
   mkrtb[20],  % marker table %                                         0526
   filhde,    %end of the file header%                                  0527
%...output sequential/quickprint...%                                    0528
   sfpmr1,  %r1 for pmaps to output file%                               0529
   sfpmr2,  %r2 for pmaps to output file%                               0530
   sfpmr3,  %r3 for pmaps to output file%                               0531
   sfbyte,  %number of bytes in file%                                   0532
   sfbufl,  %length of output buffer ( in bytes)%                       0533
   sfucf,   %if true, convert to upper case%                            0534
   sfpgno,  %if true, paginate; contains current page number%           0535
   sflnel,  %number columns (characters) per line%                      0536
   sfndlv,  %number characters per level to indent%                     0537
   sfmxnd,  %maximum number characters to indent%                       0538
   sfndbf,  %bytptr to end of op buffer%                                0539
   sfbptr,  %byte pointer to current poosition in buffer%               0540
   sflnpg,  %number of line, this page%                                 0541
   oqnhfg,  %put only page # on output quickprint if true%             01087
   oqapfg,  %try open append for O Q if true%                           0542
%...High segment page map table -- see (seqfil, processor)%             0543
   freemap[freesz],                                                      0544
%...other work areas...%                                                0545
   mswsit, %cell to accumulate time for wsi%                            0547
```

```
   mswsic, %cell to accumulate count for wsi%                          0548
   %wsd-meas cells%                                                    0549
      ldrfct, %Number of calls on lodrfb%                              0550
      ldrnct, %number of calls on lodrng%                              0551
      ldsdct, %Number of calls on lodsdb%                              0552
      fchsct, %numbr of calls on fchsdb%                               0553
      strttm, %JOBTM at start of NLS (intmeas)%                        0554
   msaddr[15],  %for measurement -- 3 * number of entries%             0555
   mslst[39],  %for measurement -- msentl * number of entries%         0556
   mslste,  %end of measurement list%                                  0557
   msrlsv,  %save area for rl during measurement%                      0558
   msr2sv,  %save area for r2 during measurement%                      0559
   msr3sv,  %save area for r3 during measurement%                      0560
   measnt,  %current item being measured%                              0561
   iswork[4],   %insert sequential work area%                          0562
   oprwrk[13],  %output processor work area%                           0563
   gtjfnt[10],  %for long getjfn calls%                                0564
   typend,                                                            01081
%...global display/print parameters...%                                0567
   %...display global support...%                                      0568
      lastch,                                                          0574
      % mkrbuf,          pointer to marker display buffer%            0575
      mkrend,          %ditto%                                         0576
      mkrptr,          %pointer into marker display buffer%            0577
      mkrcnt,          %char count of last marker found%               0578
      mkrflg,          %true if found a marker in a statement%         0579
      % cdctrn, %                                                     0580
      cdchrl,                                                          0581
      cdchct,                                                          0582
   %...fast create display support...%                                 0590
      aulsrt,  %address, current auxiliary display table%              0591
      aulsize,  %size, in lsrtl entries, of auxiliary table%           0592
      rttop,   %starting address, current table being formatted%       0593
      rtsize,    %size, current table being formatted%                 0594
   %...display support...%                                             0595
      % aplint, %                                                      0596
      gapcol,                                                          0597
      gapcnt,                                                          0598
      gappp,                                                           0599
      gapcc,                                                           0600
      cc,                                                              0601
      rt,                                                              0602
      rte,                                                             0603
      art,                                                             0604
      arte,                                                            0605
      oldlsrt,  %address of old Line Segment Reference Table%          0606
      commands[65], %first block of display commands--others to be
      dynamically allocated%                                           0610
      dpcmblk, %address of commands block currently being filled%
                                                                       0611
      dgstl,    %current character count for statement formatting%
                                                                       0612
      dgstml,   %max character count%                                  0613
      dgstid,    %stid of statement being formatted%                   0614
      dgbufe;                                                          0615
%...typewriter print variables...%                                     0621
```

```
    DECLARE EXTERNAL                                                    0622
    prmkrf = 0,     %typewriter print marker flag%                      0623
    cdpagf = 1,     %page flag%                                         0624
    pageno,     %page number for print group%                           0629
    slshfg,     %slash command flag%                                    0632
%...general global variables...%                                        0633
    bfilno, %Contains file number on bad file SIGNAL%                   0636
    ercall, %Contains the address of last call to error%                0637
    ermark, %Contains value of mark at last call to err%                0638
    fastname = 1, %Flag for which jump name algorithm to use on normal
    jumps%                                                              0639
    carpos,         %position of carraige (0=left margin)%              0647
    slashflg = 0,  %show selections flag%                               0652
    srchtype,         %Type of item to search for (name/word) contents%
                                                                        0656
%...Library  stuff...%                                                  0664
    oldflg = 0, %sticky state of libflg%                                0842
    libflg = 0, %non-zero means routine running using NLS as library
    (so don't expect input)...number indicates which%                   0665
    liblod = 0, % TRUE if load average cut-off set by hand %    01050
    jnlprog = 0, % TRUE if journal userprog in; FALSE if not. % 01051
    autostrt = 0, %True if the library started automatically at system
    startup%                                                            0666
    libmax = 4, %max value%                                             0667
    slnkrv = 1,                                                         0668
    utiltv = 4,                                                         0669
    runlib = (0, %0 if lib will not be run automatically...otherwise
    flag number to check.  Indexed by libflg%                           0670
       2, %slinker%                                                     0671
       0,                                                               0672
       0,                                                               0673
       3), %Utility%                                                    0674
%...processor variables...%                                             0675
    prseqwk,          %address of  processor's seqgen work area%        0676
    opbp,    % byte pointer to start of current string %          01004
    opcbp,   % byte pointer to current char position %           01005
    opccpos,          % current char postion %                    01006
    opcmax,  % max chars in current string %                     01007
    oplev,   % current statement level %                         01008
    opnewst,          % flag, used to make level find efficient % 01009
%...substitute variables...%                                            0677
    subcnt;     %number of substitutions made%                          0678
%...Print Journal...%                                                   0780
    DECLARE EXTERNAL                                                    0781
       pjrbab, % save for contents of rubabt %                          0782
       pjusqc, % save for pntr to user seq gen %                        0783
       pjsavf, % save for user seq gen viewspec %                       0784
       pjsw, pjlsw, % pntrs to seq workareas %                          0785
       pjrubout; % set TRUE on RUBOUT %                                 0786
    DECLARE EXTERNAL TEXT POINTER pjstid;                               0787
                                                                        0788
DECLARE EXTERNAL patch[40], patche; % patch space %                     0801
                                                                        0802
% page aligned output buffers %                                        0846
  PAGE 140B; %get aligned on page boundary (assume load starts 140B)%
                                                                        0847
```

```
    DECLARE EXTERNAL msbuff[511],  %measurement file buffer%          0848
       msbufe;  %end of measurement buffer%                           0849
    DECLARE EXTERNAL sfbuff[511],  %Sequential file buffer%           0850
       sfbufe;  %end of Sequential file buffer%                       0851
FINISH of bdata                                                       0817
```

BGS

< NLS, BGS.NLS;9, >, 29-OCT-74 10:57 KIRK ;;;;
JOURNAL: unclassified bugs 02
The copy directory command does not work for the "For File" option if some other option is specified after the file. -- (KIRK, DVN) 097
18-OCT-74 0829-PDT PLACKO at OFFICE-1: Set NNLS file private command
Distribution: FEEDBACK AT ARC, hopper at arc, placko
Received at: 18-OCT-74 08:30:22 095
I just tried to set one of my files private this morning (complete with what I believe to be the proper access list "AccessList: MAP2;") and the response i received was "not implemented". Double check please.
-- Mike 096
NDM 18-OCT-74 15:27 2h2h9
LP Problems
Message: I'm on the Delta-Data--Line-Processor via the high-speed line and TTP to ARC running the running version of NLS. I have a

statement numbers, blank lines on.

I've been getting the error message "Illegal number of blanks requested in CLINE". Also, when I do an edit which shortens a statement, it doesn't erase the line which should then be blank (e.g. the line above the one which was blank before the edit).
*****Note: [ACTION] *****

094

RLL 23-OCT-74 23:08 24285
bug: playback record/ calculator
Message: In using the playback command with the calculator, I happened to bug an invalid number. The system respondded wth that message. Left my playback hangingg. I tried contrrl O. this cause the system to loop on '?'. additional tO did not good (waited 5 minutes wall time). This happened twice so is repeatable. In future I will be more careful but I thought to would free me from playback.
*****Note: [ACTION] *****

092
088
089
action
info
25-OCT-74 1130-PDT LEE at SRI-ARC: getting out of print journal
Distribution: FEEDBACK, feedback at arc
Received at: 25-O?T-74 11:30:24
090
I do not appreciate having tc control c to get out of print journal...
091
HGL 17-OCT-74 17:53 24241
A test of Journal delivery
Message: Does it work?
*****Note: [ACTION] *****

083

JBP 17-OCT-74 14:23 24238
load file: system file error
Message: i tried several times to load file a nls file from another directory but each time i received the message "system file error", i then tried to get the file by jumping to the file by giving the dump address command the file name in link form, this worked fine.
--jon.

*****Note: [ACTION] *****
082

14-OCT-74 2143-PDT KELLEY: USER: Show Viewspecs
Distribution: MAYNARD, feedback
Received at: 14-OCT-74 21:43:05 076
(and I presume Show Leveladjust) still mention prompting. 077
14-OCT-74 2159-PDT KELLEY: playback session
Distribution: MAYNARD, feedback
Received at: 14-OCT-74 21:59:02 078
I thought CTRL-0 was supposed to work to stop the playback. Is this a bug or is there some other way to stop it without resetting and
before the end of the session. 079
KIRK 15-OCT-74 17:24 24225
Jump File return across horizontal split makes 'fst entry nonexistant'
Message: I had done several cross-file edits and was trying to update the file in the bottom window but got "fst entry nonexistant" message. When I deleted the bottom window and reloaded the file, the update command worked ok.
*****Note: [INFO-ONLY] *****
075

RLL 14-OCT-74 09:52 24214
bugs: illegal statemet return ring....
Location: (MJOURNAL, 24214, 1:w)
*****Note: [ACTION] *****
067
Comments: This occurred on Sunday, 13 Oct 74 066
Just got the same old bug 'illegal statement return ring in copysrring.' 069
This occurred after the Jump File Return command and several spaces (to step through the list). 070
Other facts: I did not cycle around and I did use slit screens during the NLS session. 071
Also, immediately after this, I find myself unable to jump to any spot on the screen. The JUmp command gives the message 'file numbers to not match in storesring'. This also happened for other jump commands. Had to do a reset. 072
12-OCT-74 2253-PDT BAIR: Bugs in running NLS8
Distribution: FEEDBACK, FEEDBACK AT OFFICE-1, maynard, watson
Received at: 12-OCT-74 22:53:01 060
A feW bugs:
The Journal doesn't work: the whole process seems to work, but the stuff is never seen again.
--The split screen still results in error msgs such as "Bad statement identifier", "no such numbers in storesring", etc., which requires that your fc and reset, and start over.
PC flags are being set without a PC locking files, particularly the Message.txt»
in Sendmail: there's no no accepted in Send the MailY/N?. 061
No frontendfor ident archived?
If the user forgets the soursethe source in Sendmail, the system says end of file exceeded". 062
These are kinda fast, but better than not... 063
JBP 14-OCT-74 10:20 24215
RFC Number Assignment in new sys~em

Message: Using preview on Office--1 i was unable to preassign an rfc number. The sendmail command reserve rfc asks some questions, one of which is "insert the number list" which i believe is a strange way of asking if the number is to be inserted into the text of the document. It is strange that this question is ask even if the document is offline, further the question indicated that it can be answered yes or no but the no answer is not accepted.
--jon.
*****Note: [ACTION] *****

059
Responded as to purpose of the command 074
KIRK 11-OCT-74 22:43 24202
file numbers do not match in storessring
Message: This occurs immediately after I did an Update Compact on xhelp. Could have something to do with new update compact stuff?
*****Note: [ACTION] *****

058
KIRK 11-OCT-74 17:06 24201
How I got an illegal memwrite
Message: I had the following programs loaded:
<KELLEY>XLEVADJUST.PROC-REP, and MOUSE. I had about 300 pages of files on my file-return ring. I loaded the IDENTFILE and made a change to BGS and BUGS records. (I enabled to get write access). I think the system had 1000 pages at the time. I did an update Old and got the message. "NLS System error". I did a verify file and the illegal memwrite occurred. I have had this happen in the past but only when running user-programs on large files. It is not a problem with the programs. It appears to be a problem with buffer space allocations.
*****Note: [ACTION] *****

057
DVN 9-OCT-74 09:29 24172
Journal Confounds Bugs with Dreams
Message: I am a member of a group exploring possibilities of controling dreams. One of the techniques is to tell anyone who appears in your dreams about the dream. Recently Elizabeth Michael appeared in some of my dreams. Since she was travelling and I had to send her some information about demonstration files anyway, I reported my dream to her in a journal item. In one of its rare moments of humour the journal gave the same name (24170,) to two items, the dream and an item by Jeanne Beck reporting a bug. The dream has the higher version number so people loading the item get the dream. Try (jjournal,24170.nls;1,1:w) if you want to learn about the bug.
*****Note: [INFO-ONLY] *****

04
3-OCT-74 0840-PDT LEE: bugs in nls
Distribution: FEEDBACK, FEEDBACK AT OFFICE-1
Received at: 3-OCT-74 08:40:42 05
I was using nls at arc this morning and a couple of thins didn't work...
I used 3 .e for an address intendig for the text to go at the end of statement
3 but it went after the first character of statement 3 instead. 06
Hope this helps - TNLS of course... 07
(J24121) 2-OCT-74 15:37: /FDBK([ACTION]) RLL 010

While in NLS I had a view that had two differenct statements repeated. Did a viewspec 'f' but it was not corrected. Tried a jump command, but the repetition was still there. Tried a viewspec 'f' and 'F' several times al wth no effect. Finally cleared it by using viewspec 'm'and 'f'. Maybe atasker problem?
RLL 011

TNLS viewspec y problems 012
RSR 8-OCT-74 15:13 31139
viewspec y
Message: viewspec little-y doesn't give me any blank lines. Tnls preview 8 Oct.3:15 pm.
*****Note: [ACTION] *****
013

Then I did Set Viewspecs y and did a Print File but there were no blank lines between statements. (SRL) 014
1-OCT-74 0539-PDT JERNIGAN: Viewspec Problem
Distribution: FEEDBACK, jernigan
Received at: 1-OCT-74 05:39:57 015
Tried printing files this morning (Tues, 10/1, 0840) with viewspec y
and cannot get anything but viewspec z. I used both Print Branch with
viewspec nyw, and myw, and also used the Set Viewspecs y command
and neither worked. 016
Has it been changed? 017
Thanks. 018
Mil. 019

to be fixed 021
NDM 3-OCT-74 17:24 24158
Output Processor Bug: SNFRel
Message: When SNFRel=On then the directive SNFShow=<=5 is interpreted wrongly, as SNFShow=>5 .
I am waiting to release the format library until this is fixed.
*****Note: [ACTION] *****
025

[HGL] HELP bugs 026
19-JUN-74 0905-PDT BAIR: Help on Jumps.
Distribution: VANNOUHUYS, FEEDBACK, bair, kelley
Received at: 19-JUN-74 09:05:12 027
after one has used the more command to see the rest of the list, how does he get back to the original display of the list...? 028
You getback to the start of the menus by doing backs. We should go to start of frames, but this is not easy to do. Solution later.-- HGL 029
JMB 18-JUL-74 14:47 23638
Help BUG--in DNLS, in work
Message: Help gives you a message to use the MORE command when there's nothing else to menu, i.e, display says "empty" if you do say "More". Try Showing: Useroptions Show All. What is the reason for this?
*****Note: [ACTION] *****
030

KIRK bug with loading non nls files. Creates a partial copy and messes up the fdb. Should be smarter, send a message and not load

non nls files. 033
these bugs will probably not get fixed before going to OFFICE 1 03
KIRK 19-SEP-74 21:53 24006
Output Journal bug
Message: Control-O in output journal quickprint says "Error" and seems to leave <PRINTER>(IDENT)FILENAME.1;1 open so that a second attempt at quickprint results in the message "<PRINTER>(IDENT)FILENAME.1;1 is busy" (not repeatable)
*****Note: [ACTION] *****
034
Backspace gives no feedback after typing the initial space in TNLS Terse Recognition mode. (KIRK) 036
NDM 17-SEP-74 16:42 31068
Output COM Problems
Location: (HJOURNAL, 31068, 1:w)
*****Note: [ACTION] *****
037
I tried Output Com Test <CA> at Office-1 in Preview, and got the error message "No such directory". We do use this command. 038
What I really want is the ability to specify a file and FTP the print file to our printer. The command, even if it did work, is not to useful to us now. 039
By the way, does Output Com <CA> work yet, or does it still put the COM file in the printer directory? 040
[CHI] 17-JUN-74 1851-PDT NORTON: output printer via sendprint to txtfile trouble
Distribution: FEEDBACK, norton
Received at: 17-JUN-74 18:51:13 041
I have put out 2 journal directivized (haha) files thru the above process and find that consistently the footers are bumped to a separate page and then the proper page restore after them. hmm something wrong with either sendprint, new output printer or me. Think its sendprint. see me for details Please>? Jim 042
look into funny prompting/? in copy sequential 043
funny ? response after typing OPTION still have [] around command-words. Need to finish coding ccopseqfil. where is inseqh and what does justified mean? 044
[KEV?] 22-SEP-74 01:27 KIRK 24027
Questionmark puts angle brackets around funny things.
Message: Somtimes questionmark puts angle brackets around things that should not have them (for example, level adjust in the Move command questionmark contains <LEVEL-ADJUST>, <[VIEWSPECS]> ... These should be changed to correspond to the rest of the questionmark messages and to Help conventions. i.e. without the angle brackets and OPTION instead of square brackets.
*****Note: [ACTION] *****
045
3-MAY-74 1622-PDT LEE: same statement displayed twice
Distribution: FEEDBACK
Received at: 3-MAY-74 16:22:37 046
I don't know why but twice today the same statement has been displayed twice in succession. Viewspec f fixes it but it does look strange. 047
CHI SUSPECTS THAT THIS IS CAUSED BY SAME BUG THAT CAUSE CORE DUMP IN TTY WINDOW. 048
Line processor bug. Tty simulation window only works if it is the

bottom window of a horizontal split. That is tty simulation does not work for a vertically split screen. 049
KIRK 19-SEP-74 23:04 24008
Bug with viewspecs in jump to name command
Location: (HJOURNAL, 24008, 1:w)
*****Note: [ACTION] *****
050
With split screens, Jump to name allows you to bug any word in any window and then searches for the name in the window where the OK is hit */which is as it should be except maybe when you are tryin to load the file into an empty window that way). The bug occurs when you happen to bug the word in a different window from the one containing the file you wish to search. It does the search properly, but applies the viewspecs from the window containing the bugged word changing the viewspecs in the window containing the file where the search took place and was displayed. 051
KIRK 23-SEP-74 16:34 24036
Bug in load program
Message: It got the .subsys; file with the same name as the one I was trying to load only in the <programs> directory. It got the .cml; file from my directory. I used altmode after typing three letters. The name of the program is FORMAT. The proper versions are in my directory, shouldn't it look there first?
*****Note: [ACTION] *****
052
RLL 24-SEP-74 13:29 24039

Message: Ken:It has happened again. Apparently the PC bit in my archivedirectory file has been set. It is just like the message.txt probllem but this time, since I don(t need to nlsize the directory file, it causes no problem. I notice it becuase the show directory command gives a file not online message in square brackets. this is what makes me think the bit was set. could this also be the reason I get a f le not online message when I enter NLS? % No % Robert L.
*****Note: [INFO-ONLY] *****
054
6-AUG-74 0934-PDT LEAVITT: wierdness in interrogate
Distribution: FEEDBACK
Received at: 6-AUG-74 09:34:18 055
sometimes when you interrogate a file without giving a specific extension, the response will be ALL the versions of that file that are archived. then, i assume, you go back and interrogate specifically the version you want. when i do that, i type INT FILE NAME & EXT, and a command accept. a couple times i sat there like a dummy waitnG FOR SOMETHING TO HAPPEN, nothing did, then typed another CA, and was quickly given the usual response, file archived on tapes blah and blah, do you WAnt it b
k. that shouldnt be there, that extra CA, if that is what is going on. could someone please explain it to me or fix it? 056
these are not bugs 093
fixed 064
14-OCT-74 2042-PDT KELLEY: playback record
Distribution: MAYNARD, FEEDBACK
RecevED AT; 14-OCT-74 20:42:57 080
won't take n for ANSWER ? . ! (a bug similar to Interrogate's

send the mail). 081
KIRK 26-SEP-74 16:41 24069
Bug in Journal citations
Location: (JOURNAL, JRNL22, J24069:gw)
*****Note: [ACTION] *****
031
Comments: Why did I get this link to a journal message, why
doesn't the link work, and who is responsible for fixing it? 032
JBP 30-SEP-74 13:38 24097
lost messages
Message: thursday or friday i sent several messages (i thought) to fdbk and to jake, however, nothing showed up in my author branch, so either i cant
figure out how to send a message or some messages got lost.
--jon.

*****Note: [ACTION] *****
024

JAKE 2-OCT-74 10:05 24119
Problems inserting invisibles in Process Command forms
Message: In trying to build a process command form containing command-accepts, some very weird things were happening. It was virtually impossible to put a space next to a command-accept - the space ended up in some other spot or else the command-accept shifted position. All command-accepts were inserted after inserting a control-v and I was working in dnls. There have been problms in the past with invisibles doing weird things but they were rarely used. However, since people will now be using them more in process command forms, thought someone might want to investigate this. Jake

*****Note: [ACTION] *****
023

KIRK 3-OCT-74 21:52 24166
Bug with print while simulating TNLS
Message: In both tasker and Line processor, printing a large amount of text with viewspec w causes a bunch of what look like form-feeds.
*****Note: [ACTION] *****
022

RSR 5-OCT-74 17:26 31123
problem with preview viewspec
Message: In tnls preview 5:15 pm pacific saturday 5 Oct. Viewspec w is not working for me it skips a page full of lines between statements, and only prints first lines. Have you changed the viewspecs on me? If so please send me a new cue card or viewspecs card or whatever. If not, what gives?
*****Note: [ACTION] *****
020

KIRK 21-SEP-74 22:03 24024
set tty command prompts T/A:
Message: needs to be changed.
*****Note: [ACTION] *****
053

NDM 10-JUL-74 12:41 23571
Subsystem with Quit-continue Question
Message: Why is it that when I Quit Nls, then continue, I'm always returned to the Editor instead of the subsystem I was in?

*****Note: [ACTION] *****

035

JAKE 12-OCT-74 17:04 24205
Problems with sendmail
Message: I cannot send a journal when logged into my own directory <feinler>. I get an error message that states: <feinler>/send-mail/.jake;1 is not an NLS file. I do not know what this means. Have tried sending mail with both null and () delimiters, so do not think it is related to that problem. Can someone look into this please? Thanks, Jake
*****Note: [ACTION] *****

073

13-OCT-74 0242-PDT KELLEY: Leveladjust and Viewspecs on and off
Distribution: IRBY, MAYNARD, feedback
Received at: 13-OCT-74 02:42:23 065
Charles, after one of the more recent nls conventions meetings, you mentioned levels and viewspecs on and off commands would go away and the defaults would therefore, be on. I took the commands out of the documentation but notice that they are still around. Should I put the documentation for them back?

066

B INTNLS

(MLK)BINTNLS (MLK)BINTNLS (MLK)BINTNLS (MLK)BINTNLS (MLK)BINTNLS
(MLK)BINTNLS (MLK)BINTNLS (MLK)BINTNLS (MLK)BINTNLS (MLK)BINTNLS

```
< NLS, BINTNLS.NLS;9, >, 16-OCT-74 12:56 DSM ;;;;
FILE bintnls  %  L10 to <rel-nls>BINTNLS %% (L10,) (rel-nls,BINTNLS,) %
                                                                        02
%......Declarations......%                                              03
   REGISTER                                                             04
      a1 = 12, %working register%                                       05
      a4 = 15, %working register%                                       06
      r1 = 1,  %working register%                                       07
      r2 = 2,  %working register%                                       08
      r3 = 3,  %working register%                                       09
      r4 = 4,  %working register%                                      010
      r5 = 5,  %working register%                                      011
      rp = 11, %record pointer%                                        012
      p = 7,   %pattern stack%                                         013
      s = 9,   %general call stack pointer%                            014
      m = 10;  %mark stack pointer%                                    015
   EXTERNAL binit, bqt;                                                016
%.....Entry points.....%                                                017
   (dex) PROCEDURE; %start DEX%                                         018
      !JSP r1,l10init;                                                  019
      initback($dxin);                                                  020
      LOOP !haltf;                                                      021
      END.                                                              022
   (dxin)PROCEDURE; % Roll in dxctl file %                             0456
      LOCAL proc, result[10]; % Address of DXSTRT procedure in the user
      program. %                                                        0457
      LOCAL TEXT POINTER tp;                                            0458
      LOCAL STRING dxname[20];                                          0459
      REF proc;                                                         0460
      % Roll in DEX user program. %                                    0461
         % Increase buffer size to 30 pages. %                         0462
            IF NOT gpbsz( 30 ) THEN                                     0463
               BEGIN                                                    0464
               btypestr($"
               Cannot set program buffer for DEX system!");             0465
               LOOP !haltf;                                             0466
               END;                                                     0467
         % Reset the user program buffer. %                           0468
            gpgmrst();                                                  0469
         % Load the user program with DEX code. %                     0470
            *dxname* ← "rel-nls, dxctl";                                0471
            ON SIGNAL ELSE                                              0593
               BEGIN                                                    0474
               btypestr($"
               Cannot load DEX program");                               0475
               LOOP !haltf;                                             0476
               END;                                                     0477
            getuprog($dxname);                                          0473
            ON SIGNAL ELSE;                                             0594
         % Get address of DXSTRT %                                     0478
            IF NOT ddtlookup($"DXSTRT", FALSE, % get procedure in user
            program. %: &proc) THEN                                     0479
               BEGIN                                                    0480
               btypestr($"
               Cannot find entry procedure");                           0481
               LOOP !haltf;                                             0482
```

```
            END;                                                    0483
      % Run DEX. %                                                  0484
         proc();                                                    0485
      btypestr($"
      Completed.");                                                 0488
      LOOP !haltf;                                                  0486
      END.
                                                                    0487
(quin)PROCEDURE(nlsparse); % Roll in query file %                   0602
      LOCAL proc, result[10]; % Address of QPORT procedure in the user
      program. %                                                    0603
      LOCAL TEXT POINTER tp;                                        0604
      LOCAL STRING quname[20];                                      0605
      REF proc;                                                     0606
      % Roll in QUERY user program. %                              0607
         % Reset the user program buffer. %                        0614
            gpgmrst();                                              0615
         % Load the user program with QUERY code. %                0616
            *quname* ← "programs, querysys";                        0617
            ON SIGNAL ELSE                                          0618
               BEGIN                                                0619
               btypestr($"
               Cannot load QUERY program");                         0620
               LOOP !haltf;                                         0621
               END;                                                 0622
            getuprog($quname);                                      0623
            ON SIGNAL ELSE;                                         0624
         % Get address of QPORT %                                   0625
            IF NOT ddtlookup($"QPORT", FALSE, % get procedure in user
            program. %: &proc) THEN                                 0626
               BEGIN                                                0627
               btypestr($"
               Cannot find entry procedure");                       0628
               LOOP !haltf;                                         0629
               END;                                                 0630
      % Run QUERY. %                                               0631
         proc(nlsparse);                                            0632
      btypestr($"
      Completed.");                                                 0633
      LOOP !haltf;                                                  0634
      END.
                                                                    0635
(jbackground) PROCEDURE; %start Journal background process%         0451
      !JSP r1,110init;                                              0452
      initback($jnlin);                                             0453
      LOOP !haltf;                                                  0454
      END.                                                          0455
(jnlin)PROCEDURE; % Roll in JNLDEL file %                           0489
      LOCAL proc, result[10]; % Address of JOUTIL procedure in the user
      program. %                                                    0490
      LOCAL TEXT POINTER tp;                                        0491
      LOCAL STRING jname[20];                                       0492
      REF proc;                                                     0493
      % Initialize. %                                              0494
         jnlprog ← FALSE;                                           0495
         !atljb(1); %no autologout%                                 0497
```

```
      nddtarm(); %arm ↑H to nddt in case we want it%                    0498
   % Roll in Journal user program. %                                    0499
      % Increase buffer size to 30 pages. %                            0500
         IF NOT gpbsz( 30 ) THEN                                       0501
            BEGIN                                                      0502
            btypestr($"
            Cannot set program buffer for JOurnal system!");           0503
            LOOP !haltf;                                               0504
            END;                                                       0505
      % Reset the user program buffer. %                              0506
         gpgmrst();                                                    0507
      % Load the user program with Journal delivery/utility code. %
                                                                       0508
         *jname* ← "rel-nls, jnldel";                                  0509
         ON SIGNAL ELSE                                                0595
            BEGIN                                                      0512
            btypestr($"
            Cannot load jnldel user program");                         0513
            LOOP !haltf;                                               0514
            END;                                                       0515
         getuprog($jname); % loads user program, does  not put out
         messages. %                                                   0511
         ON SIGNAL ELSE;                                               0596
      % Set flag indicating that user program has been rolled in. %
                                                                       0516
         jnlprog ← TRUE;                                               0517
      % Get address of joutil %                                       0518
         IF NOT ddtlookup($"JOUTIL", FALSE, % get procedure in user
         program. %: &proc) THEN                                       0519
            BEGIN                                                      0520
            btypestr($"
            No JOUTIL procedure:  error in DDT lookup");               0521
            LOOP !haltf;                                               0522
            END;                                                       0523
   IF libflg THEN % run journal delivery %                             0524
      BEGIN                                                            0525
      proc( libflg ← libflg .V oldflg, jdid % list of idents. %);
                                                                       0526
      btypestr($"
      Completed.");                                                    0531
      END                                                              0527
   ELSE btypestr($"
   No library functions set");                                         0528
   LOOP !haltf;                                                        0529
   END.
                                                                       0530
(btypestr) PROCEDURE (astrng); %Type a-string on tty%                  0532
   %-----------------%                                                 0533
   REF astrng;                                                         0534
   IF NOT astrng.L THEN RETURN;                                        0535
   !sout(1C1B, chbmty + &astrng, -astrng.L);                           0536
   RETURN;                                                             0537
   END.
                                                                       0538
(net) PROCEDURE; % Start NLS from the Net %                             023
   r1.LH ← 7; % first AC to save is 7 %                                024
```

```
   r1.RH ← $xacs [7]; % that's where it's to be saved %                    025
   r2 ← $xacs [15]; % AC 17 is last one to be saved %                      026
   ! BLT 1,(2); % save ACs 7-17 %                                          027
   !JSP r1,110init;                                                        028
   initback($netdsp);                                                      029
   LOOP !haltf;                                                            030
   END.                                                                    031
DECLARE submission=1;                                                      032
(netdsp) PROCEDURE; % Dispatch to HIOP or NJS %                            033
   % Declarations %                                                        034
      LOCAL reason;   % reason for dispatch %                              035
   % Save dispatch reason %                                                036
      r1 ← xacs [7]; % get xwd dispatch-reason, xxx %                      037
      ! HLRZS 1; % clean dispatch reason %                                 038
      reason ← r1; % save it %                                             039
      IF reason = submission THEN njs($xacs) ELSE hiop($xacs);             040
      END.                                                                 041
%.....NLS pre-initialization and SSAVE code .....%                         042
(bpresaveinit) PROCEDURE; %partial initialization of backend before
SSAVEing it as an object file %                                            043
   LOCAL return;                                                           044
   IF FALSE THEN                                                           045
      BEGIN                                                                046
      (bqt): !JSP r1,110init;                                              047
      return ← FALSE;                                                      048
      END                                                                  049
   ELSE return ← TRUE;                                                     050
   ON SIGNAL ELSE                                                         0549
      BEGIN                                                               0550
      IF sysmsg AND [sysmsg].L <= [sysmsg].M AND [sysmsg].L IN (1,
      200] THEN                                                           0551
         typeas(sysmsg);                                                  0552
      !JSYS 1147B; %Reset jsys-- cannot use symbol because of
      conflict%                                                           0553
       LOOP !haltf;                                                       0554
         %In case someone is foolish enough to try to contnue%            0555
      END;                                                                0556
   % high que so this stuff goes fast %                                    051
      setpriority(202B);                                                  052
   IF jdebug THEN % check journal stuff %                                 053
      LOOP                                                                054
         BEGIN                                                            055
         typeas($"Enable Journal System?? ");                             056
         !pbin;                                                           057
         CASE r1 OF                                                       058
            = 'y, = 'Y: %yes%                                             059
               BEGIN                                                      060
               jdebug ← 0;                                                061
               typeas($"es.
                  Password for Directory Journal (CR for default) =
               ");                                                        062
               *jnlpsw* ← NULL;                                           063
               LOOP                                                       064
                  BEGIN                                                   065
                  !pbin;                                                  066
                  IF r1 = EOL THEN EXIT;                                  067
```

```
                *jnlpsw* ← *jnlpsw*, r1;                                    068
                END;                                                        069
            IF jnlpsw.L = 0 THEN *jnlpsw* ← "JPD";                          070
            EXIT LOOP;                                                      071
            END;                                                            072
        = 'n, = 'N: %no%                                                    073
            BEGIN                                                           074
            typeas($"o.
            ");                                                             075
            *jnlpsw* ← "JPD";                                               076
            EXIT LOOP;                                                      077
            END;                                                            078
        ENDCASE                                                             079
            BEGIN                                                           080
            typeas($" ? (y or n)
            ");                                                             081
            REPEAT LOOP;                                                    082
            END;                                                            083
    END;                                                                    084
IF hdebug THEN % check help stuff %                                         0559
    LOOP                                                                    0560
        BEGIN                                                               0561
        typeas($"Enable Help System?? ");                                   0562
        ipbin;                                                              0563
        CASE r1 OF                                                          0564
            = 'y, = 'Y: %yes%                                               0565
            BEGIN                                                           0566
            hdebug ← 0;                                                     0567
            typeas($"es?
            ");                                                             0591
            EXIT LOOP;                                                      0577
            END;                                                            0578
            = 'n, = 'N: %no%                                                0579
            BEGIN                                                           0560
            typeas($"o.
            ");                                                             0581
            EXIT LOOP;                                                      0583
            END;                                                            0584
        ENDCASE                                                             0585
            BEGIN                                                           0586
            typeas($" ? (y or n)
            ");                                                             0587
            REPEAT LOOP;                                                    0588
            END;                                                            0589
        END;                                                                0590
% Set entry vector %                                                        085
    setbvec();                                                              086
% initialize free storage zone %                                           087
    % LATER this will have its own free storage zone -- noop for
    now                                                                     088
    makezone($dspblk, 2000B, 0, $dspbke - $dspblk-1);                       089
    %                                                                       090
% initialize sequence work areas %                                         091
    sqinit();                                                               092
% initialize file stuff %                                                  093
    filinit();                                                              094
```

```
    % initialize user program stuff %                                    095
      gpgmrst();                                                         096
    % initialize strings for ident system%                               097
      *idnamdel* ← '?, CR, EOL;                                          098
    IF return THEN RETURN                                               0100
    ELSE                                                                0101
      BEGIN                                                             0102
      backsave();                                                       0103
      LOOP !haltf;                                                      0104
      END;                                                              0105
    END.                                                                0106
%.....NLS initialization .....%                                         0107
  (initback) PROCEDURE (dispatcher); %start up the backend%             0108
    % This is the procedure which is called in order to start up
    NLS-backend%                                                        0109
    %------------------%                                                0110
    %!!!CANNOT HAVE LOCAL VARIABLES!!!%                                 0111
    IF FALSE THEN                                                       0112
      BEGIN                                                             0113
      (binit): %label inserted to avoid the first instruction in the
      procedure which verifies that the stack pointer is reasonable%
                                                                        0114
      !JSP r1, 110init;  %initialize 110 runtime environment%           0115
      dispatcher ← 0;                                                   0116
      END;                                                              0117
    % go start the world %                                              0118
      startback(dispatcher);                                            0119
    LOOP !haltf;                                                        0120
    END.                                                                0121
                                                                        0122
  (startback) PROCEDURE (dispatcher); %initialize the backend%          0123
    %-----------------%                                                 0124
    % initialization done every startup or re-startup %                 0125
      ON SIGNAL ELSE % Close files and issue halt jsys if error in
      startup %                                                         0126
        BEGIN                                                           0127
        IF sysmsg AND [sysmsg].M AND [sysmsg].L AND [sysmsg].M >=
        [sysmsg].I AND [sysmsg].L < 200 THEN typeas(sysmsg);            0128
        (strthlt):                                                      0129
          !JSYS 147B; %Reset jsys-- cannot use symbol because of
          conflict%                                                     0130
           LOOP !haltf;                                                 0131
             %In case someone is foolish enough to try to continue%
                                                                        0132
        END;                                                            0133
      %Check to see if we are to run library routines%                  0134
        libflg ← IF libflg THEN libflg ELSE checklib();                 0135
        IF libflg = 1 THEN                                              0597
          BEGIN                                                         0598
          setpriority( 202B);                                           0599
          libflg ← 11B;                                                 0600
          END;                                                          0601
      % misc. stuff %                                                   0136
        inptrf ← FALSE; %rubout flag%                                   0137
        lhostn ← hostnumber(); %save logical host number%               0138
        oprtty ← IF lhostn = utilhost THEN utiloprtty ELSE
```

```
            arcoprtty;                                                  0139
            %misc. for background, load averag checks%                  0140
               nsacktime ← 0; %time (gtad) to check if system going
               down%                                                    0141
               !sysgt(getsbn($"SYSTAT"));                               0142
               !HRLI r2,14B; lavtabad ← r2; %getab table address of
               load average%                                            0143
            tenex ← tenexverno();                                       0144
         IF NOT continue THEN % this stuff done only once %             0145
            BEGIN                                                       0146
            % initialize measurement tables %                           0147
               IF msflag THEN intmeas();                                0148
            % initialize globals about monitor tables %                 0149
               !sysgt(getsbn($"JOBTTY"));                               0150
               ttytbl ← r2.RH;  % table number %                        0151
               !sysgt(getsbn($"JOBDIR"));                               0152
               jobtbl ← r2.RH;  % table number %                        0153
            % create message fork %                                     0154
               % will need this LATER, but noop now                    0155
               IF nlmode = fulldisplay THEN inittimer();                0156
               %                                                        0157
            %get mode, system name%                                     0158
               *nlsnam* ← "BACKNLS";                                    0159
               nlstyp ← tntyp;                                          0160
            END;                                                        0161
          IF dispatcher THEN [dispatcher]() ELSE RETURN;                0162
         END.                                                           0163
                                                                        0164
%.....Initialization support routines.....%                            0165
    (backsave) PROCEDURE; % save nls as disk file %                     0166
         LOCAL jfn;                                                     0167
         LOOP % get save file name from user %                          0168
            BEGIN                                                       0169
            typeas($"save filename: ");                                 0170
            IF NOT SKIP !gtjfn(400003B6,100000101B) THEN                0171
               BEGIN                                                    0172
               typeas($" ?
               ");                                                      0173
               REPEAT LOOP;                                             0174
               END;                                                     0175
            jfn ← r1;                                                   0176
            EXIT LOOP;                                                  0177
            END;                                                        0178
         % save core image as dsk file %                                0179
            !ssave(400000B6 .V jfn.RH, 777000B6 .V 520000B, 0);         0180
               %WILL HAVE TO CHANGE THIS LATER FOR CORRECT BOUNDS%       0181
         RETURN;                                                        0182
         END.                                                           0183
                                                                        0184
    (setbvec)PROCEDURE; %set entry vector for backend%                  0185
         % comment out for now                                          0186
            entvec[0] ← $binit .V 254B9; start backend%                 0187
         entvec[3] ← $dex .V 254B9; %start DEX%                         0188
         entvec[4] ← $jbackground .V 254B9; %journal background process%
                                                                        0544
         entvec[6] ← $net [1] .V 254B9; %start from Network %           0189
```

```
   !sevec( 4B5, 7B6 .V $entvec); %set entry vector%                    0190
   RETURN;                                                            0191
   END.                                                               0192
(filinit) PROCEDURE;  %init file handling mechanisms%                 0193
   LOCAL                                                              0194
      fileno, count, fz, da, astrng, fl, fp, end, hdr, sndptr, char;
                                                                      0195
   REF sndptr, fl, fp, astrng, fz, da;                                0196
   %initialize file list%                                             0197
      nxpg ← 1;  %for lodrfb%                                         0198
      &fl ← $filst;                                                   0199
      end ← &fl + filmax*filstl;                                      0200
      DO fl ← 0 UNTIL (&fl ← &fl + 1) = end;                          0201
      &fl ← $filst;                                                   0202
   %Initialize filepart table%                                        0203
      &fp ← $filepart;                                                0204
      end ← &fp + filmax;                                             0205
      DO fp ← 0 UNTIL (&fp←&fp+1)=end;                                0206
   %initialize frozen list%                                           0207
      &fz ← fzfree ← $fzlsts;                                         0208
      end ← &fz + (frzmax-1)*frzl;                                    0209
      DO                                                              0210
         fz.fznext ← &fz + frzl                                       0211
      UNTIL (&fz ← &fz + frzl) = end;                                 0212
      fz.fznext ← 0;                                                  0213
   RETURN END.                                                        0214
                                                                      0215
(checklib)PROC;                                                       0225
   %This pprocedure cecks the system flags to see if it should try
   to start up one of the background processors automaticallyy.       0226
      If so, it tries to start it up.                                 0227
      If successful, it returns the library number for the
      processor,
      .                                                               0228
      otherwise, it may call err, or it may return 0 depending oon
      the error%                                                      0229
   LOCAL libndx;                                                      0230
   libndx ← 0;                                                        0231
   WHILE (libndx ← libndx+1) <= libmax DO                             0232
      IF runlib[libndx] AND flagut(runlib[libndx], $rstfg) THEN       0233
         RETURN(libstrt(libndx)); %start it up%                       0234
   %Make sure this was not a false start%                             0235
      !gjinf;                                                         0236
      IF rl = 0 THEN                                                  0237
         killib(); %Not logged in--kill job%                          0238
   RETURN(0);                                                         0239
   END.
                                                                      0240
DECLARE libsw1=%623701423, libsw2=516070141B3;                        0241
(libstrt)PROCEDURE(libno);                                            0242
   LOCAL STRING errstr[200];                                          0243
   %This procedure tries to login and start a background processor.
   It may do a directory connect too depending on the processor%      0244
   %Set up to catch signals%                                          0245
      ON SIGNAL ELSE                                                  0246
         BEGIN                                                        0247
```

```
        IF NOT sysmsg THEN sysmsg ← $"Error";                          0248
        *errstr* ← "Library Start Fail: ", *(sysmsg)*;                 0249
        specttyout(0,$errstr);                                         0250
        specttyout(30B, $errstr);                                      0251
        killib();                                                      0252
        END;                                                           0253
    %see if job is logged in%                                          0254
        !gjinf;                                                        0255
        IF r1 # 0 THEN RETURN(FALSE);                                  0256
    %enable login capability%                                          0257
        enablLogin();                                                  0258
    %Get dirno of user for login%                                      0259
        !stdir(0, chbmty + $"BACKGROUND");                             0260
            GOTO libfal;                                               0261
            GOTO libfal;                                               0262
        %By here, dir exists, no in r1.rh..see if login oK%            0263
        IF r1.b0 THEN                                                  0264
            (libfal):                                                  0265
                BEGIN                                                  0266
                specttyout(0, $"Illegal Library User");                0267
                specttyout(30, $"Illegal Library User");               0268
                killib();                                              0269
                END;                                                   0270
    %log him in%                                                       0271
        r1 ← r1.RH;                                                    0272
        r2 ← 440788 + (CASE lhostn OF                                  0273
        =utilhost: $libsw1;                                            0274
        ENDCASE $libsw2);                                              0275
        IF NOT SKIP !login(r1, r2, 5B11 .V 1) THEN                     0276
        BEGIN                                                          0277
        *errstr* ← "Could Not Log In Background Job", EOL,
        STRING(r1);                                                    0278
        specttyout(0, $errstr);                                        0279
        specttyout(30, $errstr);                                       0280
        killib();                                                      0281
        END;                                                           0282
    autostrt ← TRUE;                                                   0283
    %Make it so we are not auto-logged out%                            0284
        !atljb(1);                                                     0285
    RETURN(libno);                                                     0286
    END.
                                                                       0287
(killib)PROC;                                                          0288
    !lgout(-1); %log job out%                                          0289
    END.
                                                                       0290
(enablLogin)PROC;                                                      0291
    %Enable the Login capability%                                      0292
    !rpcap(4B5);                                                       0293
    IF NOT r2.bit3 THEN SIGNAL(0, $"Log Not in Assigned
    Capabilities");                                                    0294
    r3.bit3 ← 1;                                                       0295
    !epcap;                                                            0296
    RETURN;                                                            0297
    END.
                                                                       0298
```

```
%measurement stuff%                                                    0299
   SET                                                                 0300
      begloc=0, endloc=1, begins=2, endins=3,                          0301
      measmx=4, measct=5, frstclk=6, totclk=7;                         0302
   SET ZERO = 0;                                                       0303
   (intmeas) PROCEDURE;  %measurement initialization%                  0304
      LOCAL curitm, jsrbins, jsreins, count;                           0305
      REF curitm;                                                      0306
      %wsd-meas%                                                       0307
         !JSYS jobtm;                                                  0308
         strttm ← r1;                                                  0309
         ldrnct ← ldrfct ← ldsdct ← fchsct ← 0;                        0310
      &curitm ← $mslst;                                                0311
      jsrbins ← 264B9 .V $begmeas;                                     0312
      jsreins ← 264B9 .V $endmeas;                                     0313
      count ← 0;                                                       0314
      DO                                                               0315
         IF (curitm.msbegin ← msaddr[count]) THEN                      0316
            BEGIN                                                      0317
            IF NOT (curitm.msend ← msaddr[count ← count + 1]) THEN
                                                                       0318
               BEGIN                                                   0319
               dismes (2, $"Measurement begin address has no
               corresponding end address");                            0320
               EXIT LOOP;                                              0321
               END;                                                    0322
            curitm.msbinstn ← [curitm.msbegin] := jsrbins;             0323
            curitm.mseinstn ← [curitm.msend] := jsreins;               0324
            BUMP curitm.msbegin, curitm.msend, count;                  0325
            curitm.msmax ← msaddr[count := count + 1];                 0326
            curitm.mscurr ← curitm.mslclk ← curitm.mstclk ← 0;         0327
            END                                                        0328
         ELSE count ← count + 3                                        0329
      UNTIL (&curitm ← &curitm + msentl) >= $mslste;                   0330
      RETURN;                                                          0331
      %the following assembly code is executed when the measurement
      routines are called%                                             0332
         %this routine records the clock time to begin measurement%
                                                                       0333
         (begmeas): !ZERO;                                             0334
         !MOVEM r1,msr1sv;  %save r1, r2, r3%                          0335
         !MOVEM r2,msr2sv;                                             0336
         !MOVEM r3,msr3sv;                                             0337
         !MOVEI r1,mslst;  %index to possible measurement entry%
                                                                       0338
         (bmsfnd): !MOVE r2,begloc(r1);                                0339
         !HRRZ r3,begmeas;                                             0340
         !CAMN r2,r3;                                                  0341
         !JRST bmstim;  %this is the entry%                            0342
         !ADD r1,msentl;                                               0343
         !CAIGE r1,mslste;                                             0344
         !JRST bmsfnd;                                                 0345
         !MOVE r1,=255B9;  %entry not here -- fake return%             0346
         !MOVEM r1,bmsrin;                                             0347
         !JRST bmsrtn;                                                 0348
         (bmstim): !MOVEM r1,measnt;                                   0349
```

```
    !MOVE r1,=-5;                                                   0350
    !JSYS jobtm;                                                    0351
    !MOVE r3,measnt;                                                0352
    !MOVEM r1,frstclk(r3);                                          0353
    !MOVE r2,begins(r3);                                            0354
    !MOVEM r2,bmsrin;                                               0355
    (bmsrtn): !MOVE r1,begmeas;                                     0356
    !HRRM r1,bmsrloc;                                               0357
    !MOVE r1,msr1sv;                                                0358
    !MOVE r2,msr2sv;                                                0359
    !MOVE r3,msr3sv;                                                0360
    (bmsrin): !ZERO;                                                0361
    (bmsrloc): !JRST 0;                                             0362
%this routine records the clock time and performs the
necessary calculations at the end of the measurement
interval; to read the measurement at the end of the
specified number of passes, two breakpoints may be set              0363
   at msmbrk: at this point r1 will contain the elapsed
   time in milliseconds;                                            0364
   at mssbrk: at this point r1 will contain the elapsed
   time in seconds, and r2 the remainder, in milliseconds%
                                                                    0365
    (endmeas): !ZERO;                                               0366
    !MOVEM r1,msr1sv;  %save r1, r2, r3%                            0367
    !MOVEM r2,msr2sv;                                               0368
    !MOVEM r3,msr3sv;                                               0369
    !MOVEI r1,mslst;  %index to possible measurement entry%
                                                                    0370
    (emsfnd): !MOVE r2,endloc(r1);                                  0371
    !HRRZ r3,endmeas;                                               0372
    !CAMN r2,r3;                                                    0373
    !JRST emstim;  %this is the entry%                              0374
    !ADD r1,msent1;                                                 0375
    !CAIGE r1,mslste;                                               0376
    !JRST emsfnd;                                                   0377
    !MOVE r1,=255B9;  %entry not here -- fake return%               0378
    !MOVEM r1,emsrin;                                               0379
    !JRST emsrtn;                                                   0380
    (emstim): !MOVEM r1,measnt;                                     0381
    !MOVE r1,=-5;                                                   0382
    !JSYS jobtm;                                                    0383
    !MOVE r3,measnt;                                                0384
    !SUB r1,frstclk(r3);                                            0385
    !ADDM r1,totclk(r3);                                            0386
    !AOS 1,meascr(r3);                                              0387
    !CAMGE r1,measmx(r3);                                           0388
    !JRST emscon;                                                   0389
    !MOVE 1,totclk(r3);                                             0390
    (msmbrk): !IDIVI r1,1000;                                       0391
    (mssbrk): !SETZM meascr(r3);                                    0392
    !SETZM totclk(r3);                                              0393
    (emscon): !MOVE r2,endins(r3);                                  0394
    !MOVEM r2,emsrin;                                               0395
    (emsrtn): !MOVE r1,endmeas;                                     0396
    !HRRM r1,emsrloc;                                               0397
    !MOVE r1,msr1sv;                                                0398
```

```
            !MOVE r2,msr2sv;                                                0399
            !MOVE r3,msr3sv;                                                0400
            (emsrin): !ZERO;                                                0401
            (emsrloc): !JRST 0;                                             0402
        END.                                                                0403
                                                                            0404
FINISH                                                                      0405
This code can be deleted after 1-august-74.    -- Charles                   0406
   (oldintdspda) PROCEDURE (tat, tab, tal, tar);  %set up display areas
   for display terminal (relative to position of text area)%                0407
      %sets up top six lines as control and literal feedback areas%         0408
      %text area%                                                           0409
         tatop ← tat;                                                       0410
         tabottom ← tab;                                                    0411
         taleft ← tal;                                                      0412
         taright ← tar;                                                     0413
      %lt viewspec%                                                         0414
         lttop ← tat - 4*tavinc;                                            0415
         ltbottom ← lttop + ltvinc;                                         0416
         ltleft ← tal;                                                      0417
         ltright ← ltleft + 7*lthinc;                                       0418
      %viewspec%                                                            0419
         vstop ← tat - 3*tavinc;                                            0420
         vsbottom ← vstop + vsvinc;                                         0421
         vsleft ← tal;                                                      0422
         vsright ← vsleft + 7*vshinc;                                       0423
      %name%                                                                0424
         namtop ← tat - 4*tavinc;                                           0425
         nambottom ← namtop + namvinc;                                      0426
         namright ← tar;                                                    0427
         namleft ← tar - 25*namhinc;                                        0428
      %sybsystem name%                                                      0429
         subtop ← tat - 3*tavinc;                                           0430
         subbottom ← subtop + subvinc;                                      0431
         subright ← tar;                                                    0432
         subleft ← subright - 25*subhinc;                                   0433
      %Command Feedback Line%                                               0434
         cfltop ← tat - 4*tavinc;                                           0435
         cflbottom ← cfltop + 2*cflvinc;                                    0436
         cflleft ← ltright + cflhinc;                                       0437
         cflright ← namleft - cflhinc;                                      0438
      %literal feedback%                                                    0439
         littop ← tatop - 2*tavinc;                                         0440
         litbottom ← tabottom;                                              0441
         litleft ← taleft;                                                  0442
         litright ← taright;                                                0443
      %message (tty) area%                                                  0444
         msgtop ← tat - 6*tavinc;                                           0445
         msgbottom ← msgtop + 2*tavinc;                                     0446
         msgleft ← tal;                                                     0447
         msgright ← tar;                                                    0448
      RETURN;                                                               0449
      END.                                                                  0450
```

BRECORDS

BRECORDS BRECORDS BRECORDS BRECORDS BRECORDS BRECORDS BRECO
BRECORDS BRECORDS BRECORDS BRECORDS BRECORDS BRECORDS BRECO

```
< NLS, BRECORDS.NLS;2, >, 1-OCT-74 17:30 KEV ;;;;( MICHAEL,
BRECORDS.NLS;2, ), 16-JUL-74 09:17 EKM ;
FILE brecords % L10  <REL-NLS>brecords %% (L10,) (rel-nls,brecords.rel,)
%                                                                    0368
% records for use with the directory commands %                       03
   (fdbprr) RECORD  % description of fdbprt word in fdb %            071
      flprpu[6],       % public protection bits %                     072
      flprgr[6],       % group protection bits %                      073
      flprse[6];       % self protection bits %                       074
   (fdbprb) RECORD  % description of protection field bits %         075
      flprnu[1],       % not used %                                   076
      flprli[1],       % list protection %                            077
      flprap[1],       % append access %                              078
      flprex[1],       % execute access %                             079
      flprwr[1],       % write access %                               080
      flprre[1];       % read access %                                081
   (fdbctr) RECORD  % description of fdbctl word in fdb %            082
      fdbctu[18],      % unused bits %                                083
      fdbeph[1],       % file is ephemeral %                          084
      fdbnun[11],      % more unused bits %                           085
      fdblng[1],       % this is a long file %                        086
      fdbnxf[1],       % file doesn't exist yet - unused %            087
      fdbdel[1],       % file is deleted %                            088
      fdbnex[1],       % no ext for this fdb yet - unused %           089
      fdbprm[1],       % file is permanent %                          090
      fdbtmp[1];       % file is temporary %                          091
   (fdbarr) RECORD  % description of fdbart word in fdb %            092
      flsnat[18],      % second archive tape number %                 093
      flfsat[18];      % first archive tape number %                  094
   (fdbbfr) RECORD  % description of fdbbkf word in fdb %            095
      fldmpt[18],      % most recent dump tape number %               096
      flarfl[18];      % archive flags  (see fdbarf record) %         097
   (fdbarf) RECORD  % description of archive flags field %           098
      fdbunu[11],      % unused bits %                                099
      fdbaar[1],       % file already archived %                     0100
      fdbadl[1],       % don't delete file after archiving %         0101
      fdbmrk[1],       % archived but not marked complete - unused %
                                                                     0102
      fdbdmp[1],       % dumped but not marked complete - unused %  0103
      fdbnar[1],       % don't archive this file %                   0104
      fdbarc[1];       % archive pending %                           0105
   (fdbcnr) RECORD % description of fdbcnt word in fdb %             0106
      flnmrd[18],      % number of times this files read %           0107
      flnmwr[18];      % number of times this file written %         0108
   (fdbbyr) RECORD  % bits within fdbbyv word in fdb %               0109
      flpgsz[18],      % size of file in pages %                     0110
      flfill[6],       % unused %                                    0111
      flbyts[6],       % byte size %                                 0112
      fldfr[6];        % default no. of versions to keep %           0113
   (dlflbk) RECORD  % directory list file block %                    0114
      dlfbnb[18],      % pointer to next file block this group %     0115
      dlfbpb[18],      % pointer to previous file block this group %
                                                                     0116
      dlfbgk[36],      % group key for this file %                   0117
      dlfbsk[36],      % sort key for this file %                    0118
      dlfbal[18],      % astr - link for this file %                 0119
```

```
    dlfbff[18],         % address of file fdb %                          0120
    dlfbpf[18],         % address of pc fdb %                            0121
    dlfbfl[18],         % status bits this entry (see dlfbst RECORD) %   0122
    dlfbcp[18],         % chain pointer for second pass %                0123
    dlfbap[18];         % astr - name of this file or its pcname %       0124
                                                                          0125
    DECLARE EXTERNAL dlfbl = 6;                                           0126
                                                                          0127
(dlfbst) RECORD  % definition of dlfbfl in dlflbk RECORD %                012
    dlstig[1],          % this entry redundant - ignore it %             0129
    dlstpc[1],          % this file is a partial copy %                  0130
    dlstnl[1];          % this is an NLS file for which a pc exists %    0131
                                                                          0132
(dlgrbk) RECORD  % directory list grouping block %                       0133
    dlgbnb[18],         % pointer to next block %                        0134
    dlgbpb[18],         % pointer to previous block %                    0135
    dlgbsc[18],         % pointer to start of data chain this group %    0136
    dlgbnu[18],         % not used %                                     0137
    dlgbgk[36];         % group key for this group %                     0138
                                                                          0139
    DECLARE EXTERNAL dlgbl = 3;                                           0140
                                                                          0141
(dlmblk)  RECORD % directory list - master blocks %                       0142
    % the first 5 fields must parallel the record blkptr %               0143
       dlmbln[11],  %length of block requested%                           0144
       dlmbal[11],  %actual length of block%                              0145
       dlmbfr[1],   %true if block is on a free list%                     0146
       dlmbpr[1],   %true if previous block is on a free list%            0147
       dlmbfl[12],  % filler %                                            0148
    dlmbzn[18],         % zone to which this master block belongs %      0149
    dlmbnm[18],         % pointer to next master block %                 0150
    dlmbfp[18];         % free storage pointer %                         0151
                                                                          0152
    DECLARE EXTERNAL dlmbl = 3;                                           0153
                                                                          0154
SET EXTERNAL            % offsets within an fdb %                        0155
    fdbctl = 1,         % control word  (see fdbctr record) %            0156
    fdbprt = 4,         % protection word (see fdbprr record) %          0157
    fdbcre = 5,         % creation time & date of original vesion %      0158
    fdbuse = 6,         % LH = last write dir. # %                       0159
    fdbver = 7,         % LH is version number %                         0160
    fdbact = 10B,       % account information %                          0161
    fdbbyv = 11B,       % see fdbbyr record %                            0162
    fdbsiz = 12B,       % byte count to end of file %                    0163
    fdbcrv = 13B,       % creation time & date this version %            0164
    fdbwrt = 14B,       % time & date of last write %                    0165
    fdbref = 15B,       % time & date of last read %                     0166
    fdbcnt = 16B,       % see fdbcnr record %                            0167
    fdbbkf = 17B,       % see fdbbfr record %                            0168
    fdbart = 20B,       % see fdbarr record %                            0169
    fdbtdm = 21B,       % time & date of most recent dump %              0170
    fdbtfa = 22B,       % time & date of first archive %                 0171
    fdbtsa = 23B,       % time & date of second archive %                0172
```

```
    fdbusw = 24B;    % user settable word %                                  0173
%Record Definitions%                                                        0174
  (card) RECORD  % substitute candidate record %                            0175
    catnc[18], %length of test string%                                      0176
    carnc[18], %length of replacement string%                               0177
    catbp[36], %byte ptr to test string%                                    0178
    carbp[36], %byte ptr to replacement string%                             0179
    canxt[18]; %next card for this initial char%                            0180
  (cmblock) RECORD %command block header information%                       0181
    cbprev[18],     %previous command blk addr%                             0182
    cbnxt[18],      %next command blk addr%                                 0183
    cmnxt[18],      %next free entry, this block%                           0184
    cmlst[18],      %last useable entry, this blk%                          0185
    cmda[18];       %da address, this usage of commands list%               0186
                                                                            0187
    DECLARE EXTERNAL cbhl = 3; %command block header length%                0188
  (command) RECORD                                                          0189
    copcode[8], cparm1[18], cparm2[6],  ctype[4];                           0190
  (corpag)  RECORD  % group allocation data page format %                   0191
    corlck[36],       %lock for this page%                                  0192
    cortim[36],       %time of last locking%                                0193
    corcpy[36];       %not used%                                            0194
  (corpgr) RECORD %one word.  core page status record.  gives status
  for a given core page for random files.%                                  0195
    ctfull[1],     %true if the page is in use%                             0196
    ctfile[4],     %file to which the page belongs%                         0197
    ctpnum[9],     %page number within the file%                            0198
    ctfroz[= ];     %number of reasons why frozen%                           0199
                                                                            0200
      %The array CORPST is the core page status table and is                0201
      made up of instances of the above record.  RFPMAX gives               0202
      the number of core pages that may contain file pages.                 0203
      The core pages are located at positions indicated by                  0204
      the array CRPGAD (core page address).  CORPST is indexed              0205
      by numbers in the range [1, RFPMAX).  The starting                    0206
      location of page k is given by crpgad[k].%                            0207
                                                                            0208
  (deliverymodes) RECORD %Record containing flags for Journal delivery%
                                                                            0209
    delol[1],  %Deliver on-line%                                            0210
    delnet[1], %Deliver Network%                                            0211
    delhc[1];  %Deliver hard copy%                                          0212
  (fhflgs) RECORD %flag word in file header%                                0213
    prvsts [1], fhunused [35];                                              0214
  (fileblockheader) RECORD %fbhdl is length%                                0215
    fbnull[36],     %unused%                                                0216
    fbind[9],     %status table index%                                      0217
    fbpnum[9],    %page number in file of this block%                       0218
    fbtype[5];    %type of this block                                       0219
       hdtyp   = header                                                     0220
       sdbtyp  = data                                                       0221
       rngtyp  = ring                                                       0222
       jnktyp  = misc (such as keyword, viewchange etc.)%                   0223
                                                                            0224
  (filstr) RECORD %File status table record.  entry length = filstl,
  max no. entries = filmax%                                                 0225
```

```
   flexis[1],                  %true: entry represents an existant file%
                                                                        0226
   flhead[9],                  %crgpad index of the file header%        0227
   flbrws[1],                  %this file in browse mode%               0228
   fllock[1],                  %file was locked by another user when
   loaded%                                                              0229
   flpcread[1],        %PC read only-- write open failed (openpc)%      0230
   flaccm[8],                  %file access mask%                       0231
   fldirno[12],        %directory number for the original file%         0232
   flnoclos[1],        % do not close this file %                       0233
   flpart[18],                 %JFN for the partial copy%               0234
   flbpart[18],        %JFN for the browse partial copy%                0235
   florig[18],                 %JFN for the original file%              0236
   flastr[18],                 %address of the file name string%        0237
   flpcst[18],                 %address of partial copy name string%
                                                                        0238
   flbpcst[18];        %address of browse partial copy name string%     0239
   DECLARE EXTERNAL filstl = 4, filmax = 25;
                                                                        0240
(isqfwa) RECORD %length is 3 words%                                     0241
   isstid[36], %last completed stid%                                    0242
   isbuff[18], %sequential file buffer address%                         0243
   islevs[18],  %location of string for levadj%                         0244
   isinfno[8], %sequential file number%                                 0245
   istatnd[8], %character that signals end of statement%                0246
   islevb[8], %number of leading blanks per level%                      0247
   islstb[8], %number of leading blanks, last statement%                0248
   isdpth[8], %depth from initial psid%                                 0249
   isfiltyp[1]; %TRUE if tenex file, else from 940%
                                                                        0250
(lhjfn) RECORD          % lefthalf bits from gtjfn %                    0251
   lhjrun[4],           % unused rightmost bits %                       0252
   lhjbl3[1],           % bit 13: always zero %                         0253
   lhjbl2[1],           % bit 12: complement of B8 in gtjfn call %      0254
   lhjbl1[1],           % bit 11: ;T given %                            0255
   lhjbl0[1],           % bit 10: account given %                       0256
   lhjb9[1],            % bit 9: protection given %                     0257
   lhjb8[1],            % bit 8: use lowest version %                   0258
   lhjb7[1],            % bit 7: use next highest version %             0259
   lhjb6[1],            % bit 6: use highest version %                  0260
   vstar[1],            % asterick for version number %                 0261
   estar[1],            % asterick for extension name %                 0262
   fstar[1],            % asterick for file name %                      0263
   dstar[1],            % asterick for directory name %                 0264
   ustar[1],            % asterick for unit %                           0265
   dvstar[1],           % asterick for device %                         0266
   lfjlun[18]           % unused leftmost bits %                        0267
   ;
                                                                        0268
(measrec) RECORD                                                        0369
   msbegin[36],  %location to begin measurement%                        0370
   msend[36],  %location to end measurements%                           0371
   msbinstn[36], %instruction displaced to begin measurement%           0372
   mseinstn[36], %instruction displaced to end measurement%             0373
   msmax[36], %maximum number of passes to measure%                     0374
   mscurr[36], %current number of passes measured%                      0375
   mslclk[36], %clock at beginning of interval%                         0376
```

```
   mstclk[36]; %total clock time, this series%                          0377

   DECLARE EXTERNAL msentl=8;                                           0378
(mrker) RECORD                                                          0269
   mkname[36], mkpsid[18], mkfix[1], mkccnt[12], mkexis[1];             0270
   DECLARE EXTERNAL mkrl = 2;                                           0271
(ojtime) RECORD %on-line delivery time format%                          0272
   ojtitt[12], %hour times 64 plus fractional part of hour (64ths)%     0273
   ojtinn[6], %number of deliveries (at ojtiii hour intervals)%         0274
   ojtiii[6], %delivery interval, 0 defaults to 1%                      0275
   ojtijjj[9]; %slinker op flag%                                        0276
(opflgs) RECORD      % format of flags word passed to OP %              0277
   opform[1],        % TRUE then send form feeds %                      0278
   opsimff[1],       % TRUE if simulate form feeds %                    0279
   opwtpb[1];        % TRUE if wait at end of page breaks %             0280
(rfstr) RECORD % Random file block status record. (=0 then
unallocated, else one of the following records).%                       0281

   rfexis[1],    %true if the block exists in the file%                 0282
   rfpart[1],    %true if block comes from partial copy%                0283
   rfnull[2],    %unused%                                               0284
   rfused[10],   %used word count for the block%                        0285
   rffree[10],   %free pointer for the block%                           0286
   rfcore[9];    %0 then not in core, else page index%                  0287

   % The table RFBS is broken into two sections each of which
   contains a block a records of the above type. The first section
   includes RNGM entries from RFBS[RNGBAS] up to and including
   RFBS[RNGBAS+RNGM-1] and contains information about the ring blocks
   in the file. The second section includes DTBM entries from
   RFBS[DTBBAS] up to and including RFBS[DTBBAS+DTBM-1] and contains
   information about the data blocks in the file. The entry
   RFBS[RNGBAS+i] may also be referenced as RNGST[i]; likewise
   RFBS[DTBBAS+i] may be referenced as DTBST[i]. The index in RFBS
   of a block is the actual page number of the block in the file. %     0288

   % Data blocks are allocated in the file starting with page DTBBAS.
    Up to DTBM data blocks may be allocated, with data blocks given
   internal numbers from 0 to DTBM-1. The array DTBST in the file
   header is the data block status table and contains a one word
   record for each potential data block. A zero entry means that the
   block does not exist in the file, otherwise the entry is as
   described in the above record defintion. A pointer to an SDB
   (PSDB) consists of a nine bit data block number in the range
   [0,DTBM) and a nine bit displacement from the start of the block.
   The variable DTBL is maintained in each file header as the current
   upper bound on allocated data blocks for that file. This is used
   to limit the search for a location for a new SDB. The variable
   DTBLST contains the index of the block from which an SDB was last
   allocated or freed. %                                                0289
(seqr) RECORD  % sequence generator work area -- sqwrkl words long %
                                                                        0320
```

```
   swcstid  [36],  % real STID of current statement %                    0321
   swlbstid [36],  % STID of statement heading last branch in the
   sequence %                                                            0322
   swstid   [36],  % "stid" of last item "port-sent" from this
   sequence -- it may be an ENDFIL, a stastr (pointer to an
   a-string), or the same as SWCSTID %                                   0323
   swusqcod [18],  % address of user sequence generator code or 0 %
                                                                         0324
   swcacode [18],  % address of content anylyzer code or 0 %            0325
   swvspec  [36],  % first word of viewspecs %                          0326
   swvsp2   [36],  % second word of viewspecs %                         0327
   swsrsav  [36],  % save area for s-register when switch stacks %
                                                                         0328
   swmrsav  [36],  % save area for m-register when switch stacks %
                                                                         0329
   swsvw    [18],  % address of statement vector work area %            0330
   swslvl   [6],   % level at which sequence was started %              0331
   swclvl   [6],   % current statement's level %                        0332
   swkflg   [1],   % has first item in this sequence been port-send
   -- used for k viewspec %                                              0333
   swalloc  [1],   % whether or not this work area is allocated %
                                                                         0334
   swcall   [1];   % is user seqgen (or SSEQGEN) to be CALLed -- or
   gotten to thru the "port-send" mechanism? %                           0335
   SET EXTERNAL sqwrkl = 9;                                              0336
                                                                         0337
(shed) RECORD  % substitute header record %                              0338
   sbrp[36], %pointer to next space for card%                            0339
   sbdp[36], %pointer to dispatcher%                                     0340
   sbas[36], %pointer to A-string for strings%                           0341
   sbtt[36]; %type of entity being substituted%                          0342
      %this field can be shortened: only needs to hold numbers up to
      32, at the outside%                                                0343
(substr) RECORD %substitute stack entry record%                          0344
   sbc1cnt [12], sbc2cnt [12], sbc3cnt [12];
                                                                         0345
(tenexbits)RECORD %Bits according to the TENEX way%                      0346
   bitfiller[32],                                                        0347
   bit3[1],                                                              0348
   bit2[1],                                                              0349
   bit1[1],                                                              0350
   b0[1];                                                                0351
(usrblk)  RECORD  %group allocation userjob entry record%                0352
   usrlnk[36],      %lh = back link to previous user block%              0353
                    %rh = forward link%                                  0354
   usrno[18],       %user directory number(rh!)%                        0355
   usrflgs[18],     %flags (see defs)(lh!)%                             0356
   usrgrp[36],      %user group number%                                 0357
   usrjob[36],      %user job number%                                   0358
   usrtty[36],      %user tty number(-1 if detached)%                   0359
   usrtod[36],      %time logged in(for connectime)%                    0360
   usrrt[36],       %runtime for this job%                              0361
   usrtim[36];      %todclk of last runtime update%                     0362
DECLARE EXTERNAL lcard = 4; %length of card in words%                    0363
                                                                         0364
DECLARE EXTERNAL lshed = 4; %length of shed in words%                    0365
```

0366
0367

FINISH

CALC SUPPORT

CALCSUPPORT CALCSUPPORT CALCSUPPORT CALCSUPPORT CALCSUPPORT CA
CALCSUPPORT CALCSUPPORT CALCSUPPORT CALCSUPPORT CALCSUPPORT CA

```
< NLS, CALCSUPPORT.NLS;7, >, 10-SEP-74 11:32 EKM ;;;;
FILE calcsupport % L10 <REL-NLS>CALCSUPPORT.rel %% (110,)
(REL-NLS,CALCSUPPORT.rel,) %                                              02
% DECLARATIONS %                                                         03
   REGISTER                                                               04
      r0 = 0, r1 = 1, r2 = 2, r3 = 3, r4 = 4, r5 = 5, r6 = 6, a1 = 12,
                                                                          05
      a2 = 13, a3 = 14, a4 = 15;
                                                                          06
   DECLARE FIELD                                                         0158
      expf=[3B,9:27];
                                                                         0159
   DECLARE                                                               0160
      fone=2014B8,                                                       0161
      ften=2045B8, ften1=151B9,                                          0162
      ftenth=1756314631468, ften11=1423146314638;
                                                                         0163
% ARITHMETIC %                                                           07
   (qcmult)PROC (ar,ma); %qfloating multiply ar * ma, result to ar%       08
      %parameters are all addresses%                                     09
      !MOVE r3,ma;                                                       010
      !MOVE r2,ar;                                                       011
      !MOVE r4,0(r2);                                                    012
      !MOVE r5,1(r2);                                                    013
      !MOVEM r4,r6;                                                      014
      !FMPR r6,1(r3);                                                    015
      !FMPR r5,0(r3);                                                    016
      !UFA r5,r6;                                                        017
      !FMPL r4,0(r3);                                                    018
      !UFA r5,r6;                                                        019
      !FADL r4,r6;                                                       020
      !MOVEM r4,0(r2);                                                   021
      !MOVEM r5,1(r2);                                                   022
      RETURN; END.
                                                                         023
   (qcadd) PROC (ar,ma); %qfloating add ar + ma, result to ar%           024
      !MOVE r3,ma;                                                       025
      !MOVE r2,ar;                                                       026
      !MOVE r4,0(r2);                                                    027
      !MOVE r5,1(r2);                                                    028
      !UFA r5,1(r3);                                                     029
      !FADL r4,0(r3);                                                    030
      !UFA r5,r6;                                                        031
      !FADL r4,r6;                                                       032
      !MOVEM r4,0(r2);                                                   033
      !MOVEM r5,1(r2);                                                   034
      RETURN END.
                                                                         035
   (qcdiv)PROC (ar,ma); %call divide and replace accum with quo%         036
      LOCAL quo[2];                                                      037
      REF ar, ma;                                                        038
      qcdivw(&ar,&ma,$quo);                                              039
      ar ← quo;                                                          040
      ar[1] ← quo[1];                                                    041
      RETURN; END.
                                                                         042
```

```
(qcdivw)PROC (ar,ma,quo);%divide ar by ma, result in quo%
   %parameters are all addresses%
   !MOVE r3,ma;
   !MOVE r2,ar;
   !MOVE r4,0(r2);
   !MOVE r5,1(r2);
   !FDVL r4,0(r3);
   !MOVN r6,r4;
   !FMPR r6,1(r3);
   !UFA r5,r6;
   !FDVR r6,0(r3);
   !FADL r4,r6;
   !MOVE r2,quo;
   !MOVEM r4,0(r2);
   !MOVEM r5,1(r2);
   RETURN END.

(qcsub)PROC (ar,ma);%qfloating subtract ar - ma, result to ar%
   !MOVE r3,ma;
   !MOVE r2,ar;
   !MOVE r4,0(r2);
   !MOVE r5,1(r2);
   !DFN r4,r5;
   !UFA r5,1(r3);
   !FADL r4,0(r3);
   !UFA ?R5,r6;
   !FADL r4,r6;
   !DFN r4,r5;
   !MOVEM r4,0(r2);
   !MOVEM r5,1(r2);
   RETURN; END.

(qcneg)PROC(ar);%floating negate double precision of ar, result in
ar%
   !MOVE r2,ar;
   !MOVE r4,0(r2);
   !DFN r4,1(r2);
   !MOVEM r4,0(r2);
   RETURN; END.

(qfloat) PROC(ar,ch); %qfloat ch (ascii char from 0-9)%
   %ar is address of result%
   LOCAL expa;
   REF ar;

   expa ← 200B;
   r3←0;
   .expf←(ch-'0);
   WHILE SKIP !TLNN r3,777000B DO
      BEGIN
      !LSH r3,-1;
      BUMP expa;
      END;
   .expf←expa;
   ar←r3;
   BUMP &ar;
```

```
   r3←0;                                                              095
   .expf←expa-27;                                                     096
   ar←r3;                                                             097
   RETURN; END.                                                       098

(nfloat) PROCEDURE (instring, fll, fl2); %convert char - dp%          099
   %convert character string to double precision floating point%      0100
   %instring is address of input string, fll, fl2 will contain
   results%                                                           0101
   LOCAL chr,fchr,fchrl,term,terml,sflg;                              0102
   REF fll, fl2, instring;                                            0103
                                                                      0104
   %Set up flag for negative umber%                                   0105
   sflg ← FALSE;                                                      0106
   %set READC pointer to head of string%                              0107
   CCPOS  SF(*instring*);                                             0108
   fll ← fl2 ← 0; %clear result area%                                 0109
   %take care of portion to left of decimal point%                    0110
   CASE (chr ← READC) OF                                              0111
      IN ['0,'9]:                                                     0112
         BEGIN                                                        0113
         qcmult(&fll, $ften);                                         0114
         qfloat($fchr,chr);                                           0115
         qcadd(&fll, $fchr);                                          0116
         REPEAT CASE;                                                 0117
         END;                                                         0118
      = '-:                                                           0119
         BEGIN                                                        0120
         sflg ← TRUE;                                                 0121
         REPEAT CASE;                                                 0122
         END;                                                         0123
      = '.:                                                           0124
         %take care of portion to right of decimal point%             0125
         BEGIN                                                        0126
         term←fone;                                                   0127
         terml←0; %temporaries%                                       0128
         CASE (chr ← READC) OF                                        0129
            IN ['0,'9]:                                               0130
               BEGIN                                                  0131
               qcmult($term,$ftenth);                                 0132
               qfloat($fchr,chr);                                     0133
               qcmult($fchr,$term);                                   0134
               qcadd(&fll,$fchr);                                     0135
               REPEAT CASE;                                           0136
               END;                                                   0137
                                                                      0138
            = '-:                                                     0139
               BEGIN                                                  0140
               sflg ← TRUE;                                           0141
               REPEAT CASE;                                           0142
               END;                                                   0143
                                                                      0144
            = ENDCHR:                                                 0145
               EXIT CASE;                                             0146
                                                                      0147
         ENDCASE REPEAT CASE; %drop editing characters%               0148
```

```
          END;                                                          0149
                                                                        0150
        = ENDCHR: EXIT CASE;                                            0151
                                                                        0152
        ENDCASE REPEAT CASE; %drop editing characters%                  0153
                                                                        0154
    RETURN (sflg);                                                      0155
    END.
                                                                        0156
FINISH                                                                  0157
```

CALCULATOR.

CALCULATOR CALCULATOR CALCULATOR CALCULATOR CALCULATOR CALCULA
CALCULATOR CALCULATOR CALCULATOR CALCULATOR CALCULATOR CALCULA

```
< NLS, CALCULATOR.NLS;6, >, 18-SEP-74 15:20 EKM ;;;;( MICHAEL,
PSCALC.NLS;1, ), 22-MAR-74 02:51 KEV ;
FILE calculator % L10 <PROGRAMS>CALCULATOR.subsys %% (110,)
(PROGRAMS,CALCULATOR.subsys,) %                                          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
      fld3[6], #chars in exp                                            023
      fld2[6], #chars right of decimal                                  024
      fld1[6], #chars to left of decimal                                025
      round[5], # significant digits to round to                        026
      blnk[1], not used                                                 027
      oflo[1], go to free format on column overflow                     028
      exsign[2], if not a negative exponent,  0 - 1st char after prefix 029
      exp2[2],   exponent prefix  0 - no exp  1 - " " " digit           030
      dpt[1],    print decimal pt 1 - "E"     2 - " " " "+"             031
      dol[1],    print $          2 - "D"                 " space      032
      dig[1],    print at least one digit # if necessary 3 - "*10^"    033
      just[2],                                  left of decimal         034
      sign[2];   on positive numbers  0 - 1st char is digit             035
                                      2 - 1st char is "+"               040
                                      1 - 1st char is space
% CALCULATOR SUBSYSTEM %
   % 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 cbadent THEN abortsubsystem($"You are already in
                  the calculator")                                      051
```

left of decimal 0 - fill with sp 1 - " " "0" 2 - " " "*" 3 - left just with spaces

```
            ELSE cbadent ← TRUE;                                    052
         % load and set up calculator file %                        053
            cafilinitialize();                                      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, $mess);                                   0100
                  % initialize literal area %                           0101
                    setlit();                                           0102
                  END;                                                  0103
            ENDCASE                                                     0104
               BEGIN                                                    0105
               % old TNLS bottom of loop %                              0106
                  %handle TNL hard copy feedback%                       0107
                    IF NOT nofeedbk THEN % long form %                  0108
                       BEGIN                                            0109
                       typeas($blanks);                                 0110
                       typeas($opstring);                               0111
                       typeas($"    ");                                 0112
                       typeas($acstring);                               0113
                       END;                                             0114
               END;                                                     0115
            %set defaults %                                             0116
               proc ← $qcadd;                                           0117
               *opstring* ← NULL;                                       0118
               *signstr* ← '+;                                          0119
               namereset ← TRUE;                                        0120
            END;                                                        0121
      ENDCASE;                                                          0122
      RETURN(&result);                                                  0123
      END.                                                              0124

                                                                        0125
% ARITHMETIC %                                                         0126
   (xcarith) PROCEDURE                                                  0127
      (result,                                                          0128
      parsemode,                                                        0129
      operation,                                                        0130
      numptr);                                                          0131
      LOCAL tp2;                                                        0132
      REF result, operation, numptr, tp2;                               0133
      CASE parsemode OF                                                 0134
         =parsing:                                                      0135
            BEGIN                                                       0136
            &tp2 ← &numptr + d2sel;                                     0137
            *opstring* ← numptr tp2;                                    0138
            CASE operation OF                                           0139
               = $add:                                                  0140
                  BEGIN                                                 0141
                  *signstr* ← '+;                                       0142
                  proc ← $qcadd;                                        0143
                  END;                                                  0144
                                                                        0145
               = $subtract:                                             0146
                  BEGIN                                                 0147
                  *signstr* ← '-;                                       0148
                  proc ← $qcsub;                                        0149
                  END;                                                  0150
               = $multiply:                                             0151
                  BEGIN                                                 0152
                  *signstr* ← '*;                                       0153
                  proc ← $qcmult;                                       0154
```

```
                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,$opfloat); %do the arithmetic and get
          answer in accum%                                              0165
          END;                                                          0166
        ENDCASE NULL;                                                   0167
      RETURN(&result);                                                  0168
      END.
                                                                        0169
% CLEAR ACCUMULATOR(S) %                                               0170
  (xclraccum) PROCEDURE (resultptr, parsemode);                         0171
      REF resultptr;                                                    0172
      CASE parsemode OF                                                 0173
        = parsing:                                                      0174
          BEGIN                                                         0175
          caclear(asub=@  ):
          case nlmode OF                                                0177
            = fulldisplay:                                              0178
              BEGIN                                                     0179
              dn($"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($o=float); %convert to floating point and
                    take care of sign %                             0257
                    [proc]($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, cafld1,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.dahinc);                    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);                        0526
   LOCAL stid, index;                                                  0527
   LOCAL STRING                                                        0528
      temstr[4],                                                       0529
      calcstr[5],    % special calc signature %                       0530
      sigstr[35],   % for checking signature %                         0531
      value[22];                                                       0532
   LOCAL TEXT POINTER start, end;
                                                                       0533
   REF resultptr;                                                      0534
   CASE parsemode OF                                                   0535
      = parsing:                                                       0536
         BEGIN                                                         0537
         %initialize locals%                                           0538
            *calcstr* ← "CALC";                                        0539
            index ← 0;                                                 0540
            stid ← orgstid;                                            0541
            stid.stfile ← cda.dacsp.stfile;                            0542
            stid ← <FILMNP, getsub>(stid); %location of info%          0543
            fechsig(stid, $sigstr);                                    0544
            *temstr* ← *sigstr*;                                       0545
            CCPOS SF(stid);                                            0546
            IF *temstr* # *calcstr* THEN                               0547
               BEGIN                                                   0548
               csysbad();                                              0549
               RETURN(&resultptr);                                     0550
               END;                                                    0551
         %check for nothing saved%                                     0552
            IF NOT FIND ["CALC ACCUMS:"] ↑start THEN                   0553
               BEGIN                                                   0554
               IF NOT FIND start [!/] <CH ↑end THEN                    0555
                  BEGIN                                                0556
                  csysbad();                                           0557
                  RETURN(&resultptr);                                  0558
                  END;                                                 0559
               END;                                                    0560
            %get and convert values%                                   0561
            DO                                                         0562
               BEGIN                                                   0563
               IF NOT FIND start [!/] <CH ↑end THEN                    0564
                  BEGIN                                                0565
                  csysbad();                                           0566
                  RETURN(&resultptr);                                  0567
                  END;                                                 0568
               *value* ← start end;                                    0569
               IF nfloat($value, $accum + index, $accum + index +
               1) THEN qcneg($accum+index);                            0570
               FIND end >CH ↑start; %get over slash delimiter%
```

```
                                                                    0571
      END                                                           0572
   UNTIL (index ← index + 2) > 19;                                  0573
%retrieve format variables%                                         0574
   stid ← <FILMNP, getsuc>(stid); %location of format
   flags%                                                           0575
   fechsig(stid, $sigstr);                                          0576
   *temstr* ← *sigstr*;                                             0577
   IF *temstr* # *calcstr* THEN                                     0578
      BE?IN                                                         0579
      csysbad();                                                    0580
      RETURN(&resultptr);                                           0581
      END;                                                          0582
   CCPOS SF(stid);                                                  0583
   IF NOT FIND ["FORMAT:"] ↑start THEN                              0584
      BEGIN                                                         0585
      csysbad();                                                    0586
      RETURN(&resultptr);                                           0587
      END;                                                          0588
   IF NOT FIND  3D ↑end THEN                                        0589
      BEGIN                                                         0590
      csysbad();                                                    0591
      RETURN(&resultptr);                                           0592
      END;                                                          0593
   *value* ← start end;                                             0594
   cacflg ← IF *value*[1] = '0 THEN FALSE ELSE TRUE;                0595
   cadflg ← IF *value*[2] = '0 THEN FALSE ELSE TRUE;                0596
   calflg ← IF *value*[3] = '0 THEN FALSE ELSE TRUE;                0597
   %retrieve format mask%                                           0598
      stid ← <FILMNP, getsuc>(stid); %location of format
      mask%                                                         0599
      fechsig(stid, $sigstr);                                       0600
      *temstr* ← *sigstr*;                                          0601
      IF *temstr* # *calcstr* THEN                                  0602
         BEGIN                                                      0603
         csysbad();                                                 0604
         RETURN(&resultptr);                                        0605
         END;                                                       0606
      CCPOS SF(stid);                                               0607
      FIND SF(stid) ↑start;                                         0608
      IF NOT FIND 1$12D ↑end THEN                                   0609
         BEGIN                                                      0610
         csysbad();                                                 0611
         RETURN(&resultptr);                                        0612
         END;                                                       0613
      *value* ← start end;                                          0614
      dfoutm ← VALUE($value);                                       0615
      %code to verify format may be inserted here - if
      user has messed with this statement, he may get a
      sudden illegal instruction eventually%                        0616
      dpset(dspno, endfil, endfil, endfil);                         0617
      END;                                                          0618
   ENDCASE NULL;                                                    0619
RETURN(&resultptr);                                                 0620
END.
                                                                    0621
```

```
   (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
      cbadent ← 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 NULL; % 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* ← "                ";                                      0897
   !MOVE r3,ar;                                                     0898
   !MOVE r2,0(r3);                                                  0899
   !MOVE r3,1(r3);                                                  0900
   r4←dfoutm; %indicator of precision%                              0901
```

```
r1 ← &os + 440700000001B; %TENEX string designator constant%
IF NOT SKIP !JSYS dfout THEN errx(whence,ar);
os.L ← os.M;
CCPOS SF(*os*);
FIND ↑tp1 $( D/ './ 'E/ '-) ↑tp2;
*os* ← tp1 tp2;
IF *os*[1] = '- THEN fst ← 2
ELSE fst ← 1;
FOR i ← fst UP UNTIL > os.L DO
   IF *os*[i] = '0 THEN *os*[i] ← SP
   ELSE EXIT;
IF cacflg THEN
   BEGIN
   j ← os.L + 2;
   FOR i ← os.L DOWN UNTIL = 1 DO
      IF *os*[i] # '. THEN
         BEGIN
         *cos*[j] ← *os*[i];
         j ← j-1;
         END
   ELSE EXIT;
   *cos*[j] ← '.;
   i← i-1;
   j ← j-1;
   LOOP BEGIN
   k ← i;
   FOR i DOWN UNTIL = k-3 DO
      BEGIN
      IF i < 1 THEN EXIT LOOP 2;
      IF *os*[i] = SP THEN EXIT LOOP 2;
      IF *os*[i] = '- THEN EXIT LOOP 2;
      *cos*[j] ← *os*[i];
      j ← j-1;
      END;
   IF *os*[i] = SP THEN EXIT LOOP;
   IF i <= 1 THEN EXIT LOOP;
   IF *os*[i] # '- THEN
      BEGIN
      *cos*[j] ← ',;
      j← j-1;
      REPEAT LOOP;
      END
   ELSE EXIT LOOP;
   END;
   IF calflg THEN BEGIN
      FIND SF(*cos*) $SP (D/'.) ↑tp1 ←tp1;
      FIND SE(*cos*) $NP ↑tp2;
      IF *os*[1] = '- THEN k ← 2
      ELSE k ← 1;
      *os*[k TO os.M] ← tp1 tp2;
      END
   ELSE
      BEGIN
      FIND SE(*cos*) ↑tp3 $NP ↑tp2;
      ST tp2 tp3 ← NULL;
      IF *os*[1] = '- THEN k ← 2
```

```
            ELSE K← 1;                                                    0958
            *OS*(K TO OS.M) ← *COS*;                                      0959
            END;                                                          0960
         END;                                                             0961
      IF *OS*(OS.I) = '.' THEN OS.L ← OS.L-1;                             0962
      RETURN END.
                                                                          0963
FINISH of CALCULATOR                                                      0964
```

CHELP

CHELP CHELP CHELP CHELP CHELP CHELP CHELP CHELP CHELP
CHELP CHELP CHELP CHELP CHELP CHELP CHELP CHELP CHELP

```
< NLS, CHELP.NLS;12, >, 26-SEP-74 07:48 HGL ;;;;
FILE chelp % 110 <rel-nls>chelp %% (L10,) (rel-nls,chelp.rel,) %        02
   % DECLARATIONS %                                                     03
                                                                     02304
      REGISTER r1 = 1;                                               03095
      REF inpt;                                                      03338
      REF conrng, qda, cda, qsw, curstk, qstorblk, qnewstmt, hlpcmdstk;
      % For help system %                                            03339
   % Initialization and display %                                    03340
      (qdisp) PROCEDURE (stid);                                      03341
         LOCAL dummy, endflg;                                        03342
         % set up the user sequence generator address for use by
         printg.The only case in which qsw is non-zero here is on
         continued menus %                                           03343
            IF NOT &qsw THEN                                         03344
               BEGIN                                                 03345
               IF NOT stid THEN RETURN(FALSE);                       03346
               % A new sequence%                                     03347
                  &qsw ← openseq(stid, stid, qda.davspec, qda.davspc2,
                  qda.dausqcod, qda.dacacode);                       03348
               % Initialize globals for queryseq %                   03349
                  qspcreset(TRUE); % default query viewspecs %       03350
                  qmenucnt ← 0;                                      03351
               qda.dacsp ← stid;                                     03352
               qda.daccnt ← 1;                                       03353
               overscreen ← FALSE;                                   03354
               END                                                   03355
            ELSE IF NOT moremen THEN                                 03356
               BEGIN                                                 03357
               % This is an error condition; if there is a sequence
               around, should not get in here unless moremen is TRUE--
               more to be shown.  Close the sequence and return FALSE %
                                                                     03358
               % Clean up the last work area if there is one %       03359
               seqgen(&qsw); % To clean up %                         03360
               qda.davspec ← qsw.swvspec;                            03361
               qda.davspc2 ← qsw.swvsp2;                             03362
               IF nlmode =fulldisplay THEN dspvsp(qda.davspec,
               qda.davspc2, 3);                                      03363
               closeseq(&qsw:=0);                                    03364
               RETURN( FALSE );                                      03365
               END;                                                  03366
         ON SIGNAL ELSE RETURN(FALSE);                               03367
         % Display node and menu. %                                  03368
            IF nlmode = fulldisplay THEN                             03369
               dafrmt(&qda, &qsw:dummy, endflg)                      03370
            ELSE                                                     03371
               BEGIN                                                 03372
               <TSPRT, printg>(&qda, stid, stid, brnchv, &qsw);      03373
               endflg ← TRUE;                                        03374
               END;                                                  03375
         % Check if finished or if end of screen or menu limit reached.
         %                                                           03376
            IF qsw.swcstid = endfil THEN                             03377
               BEGIN                                                 03378
               closeseq(&qsw:=0);                                    03379
```

```
            qda.davspec ← qdavspc;                                  03380
            qda.davspc2 ← qdavs2;                                   03381
            IF nseqer THEN                                          03382
               abortsubsystem($"Ran out of space making menus.");
                                                                    03383
            RETURN(1);                                              03384
            END                                                     03385
         ELSE                                                       03386
            BEGIN                                                   03387
            % Menu limit has been reached or end of screen hit %
                                                                    03388
               overscreen ← NOT endflg;                             03389
            qda.dacsp ← qsw.swstid ← qsw.swcstid; % so we can go
            through again. %                                        03390
            % user should be asked whether more desired-- get answer.
            %                                                       03391
               RETURN(-1);                                          03392
            END;                                                    03393
      END.
                                                                    03394
   (hlpstrt) PROCEDURE(filptr, namarray);                           04504
      LOCAL count, savstd;                                          04505
      LOCAL TEXT POINTER savptr;                                    04506
      REF filptr, namarray;                                         04507
      IF NOT namarray THEN RETURN(endfil);                          04508
      savptr ← filptr;                                              04509
      savptr[1] ← filptr[1];                                        04510
      % Find main branch %                                         04511
         lookup(&filptr, namarray[1], nametyp);                     04512
         IF filptr = endfil THEN                                    04513
            BEGIN                                                   04514
            % We don't want this to fail here; try to send the user
            off to "welcome" branch. %                              04515
            filptr ← savptr;                                        04516
            filptr[1] ← savptr[1];                                  04517
            lookup(&filptr, $"welcome", nametyp);                   04518
            RETURN(filptr); % could fail here %                     04519
            END;                                                    04520
      % Check next level in namarray.  If it isn't there we should
      start over in the "universal" branch. %                       04521
         IF namarray < 2 THEN RETURN(filptr)                        04522
         ELSE                                                       04523
            BEGIN                                                   04524
            IF (savstd ← namingrp(filptr, filptr, namarray[2],
            1000))=endfil THEN                                      04525
               BEGIN                                                04526
               filptr ← savptr;                                     04527
               filptr[1] ← savptr[1];                               04528
               namarray[1] ← $"universal";                          04529
               RETURN(hlpstrt(&filptr, &namarray));                 04530
               END;                                                 04531
            END;                                                    04532
         filptr ← savstd;                                           04533
      FOR count ← 3 UP UNTIL > namarray DO                          04534
         BEGIN                                                      04535
         IF (savstd ← namingrp( filptr, filptr, namarray[count],
```

```
      1000))=endfil THEN                                              04536
         EXIT LOOP                                                    04537
      ELSE                                                            04538
         filptr ← savstd;                                             04539
      END;                                                            04540
   RETURN(filptr);                                                    04541
   END.                                                               04542

% qstrinit should allocate a large block from dspblk, put its
address in qstorblk;  this will be used to get  smaller blocks.
xhquit will deallocate this large block if it is non-zero. %
                                                                      03421
(qstrinit) PROCEDURE (entstd, topstd);                                03422
   % Called to initialize the stack area as a storage zone for
   help/query.  Makes use of stgmgt routines to set up blocks in
   the display area block pool.  Also responsible for initializing
   some global cells as well as the context ring %                    03423
   % Allocate string %                                                03424
      &qnewstmt ← getstring( 2000, $dspblk);                          03425
   % Get context ring and initialize. %                               03426
      IF NOT (&qstorblk ← getblk(1777B, $dspblk)) THEN                03427
         abortsubsystem($"System storage problem.  Call ARC
         programmer");                                                03428
      &qstorblk ← &qstorblk + bnl;                                    03429
      makezone(&qstorblk, 70B, 70B, 1777B);                           03430
      IF NOT (&conrng ← getblk(67B, &qstorblk)) THEN                  03431
         abortsubsystem($"System storage problem.  Call ARC
         programmer");                                                03432
      conrng[1] ← &qstorblk; % first word in blocks is the address
      of the zone from which it came. %                               03433
      &conrng ← &conrng + 2; % conrng now points beyond the block
      header and the zone address to the actual stack. %              03434
      confre ← 0;                                                     03435
   % Initialize the top and entry point. %                           03436
      % For the top point, perhaps we should have two stids: stid
      of the "introduction" branch for the data base and stid of
      the "directory" beanch.  Will not need (should not have?) a
      context stack for the top!  Coming into this procedure=@
      topstd should be the stid of the origin of the data base.
      From this node, we may extract links to the "intro" and
      directory branches (which may be the same if desired.)
      CHANGE THIS CODE! %                                             03437
         topcon ← topstd; % first word is stid %                      03438
         topcon[1] ← 0; % totcnt, constk, and stkyp fields are 0 %
                                                                      03439
      entcon ← entstd; % first word is stid %                         03440
      entcon[1] ← 0; % totcnt, constk, and stkyp fields are 0 %
                                                                      03441
   % Initialize global for building entry menu %                     03442
      % The first call on pushent should get a stack if necessary
      nd put the type and address ithe context ring.  The free
      pointer should be incremented then.  %                          03443
      &curstk ← $entcon;                                              03444
      newstk ← 1;                                                     03445
   RETURN; % and then print the entry menu for help %                 03446
   END.
```

```
                                                                    03447
   (helphlp) PROCEDURE;                                             03448
      % save the accumulators %                                     03449
      svacl ← r1; r1 ← $svacs; !BLT r1, svacse;                     03450
      s ← s + 400000400B;                                           03451
      qagain ← TRUE;                                                03452
      enthelp( TRUE );                                              03453
      !HRLZI r1, svacs;                                             03454
      !BLT r1, 17B;                                                 03455
      r1 ← svacl;                                                   03456
      !JSYS debrk;                                                  03457
      END.
                                                                    03458
   (moreterm) PROCEDURE; %  Must be at a lower level because of
   symbol conflicts. Closes sequences and resets viewspecs. %       04495
      % Clean up the last work area if there is one %               04496
      seqgen(&qsw); % To clean up %                                 04497
      qda.davspec ← qsw.swvspec;                                    04498
      qda.davspc2 ← qsw.swvsp2;                                     04499
      IF nlmode = fulldisplay THEN dspvsp(qda.davspec, qda.davspc2,
      3);                                                           04500
      closeseq(&qsw:=0);                                            04501
      RETURN;                                                       04502
      END.
                                                                    04503
% Miscellaneous command control %                                  03459
% Search %                                                         03460
   (qsearch) PROCEDURE (list, index);                               03461
      % Given the address of a string containing the node list of an
      item to be found and an index to the context ring element form
      which the search will start , search for the specified node.
      Returns the stid of the found node (or -1 if not found) and an
      index to a new context ring element (or the old index if the
      requested item is not found.) SRCHTYP specifies the search
      algorithm to be used if a context search (over the context
      nodes-- very fast) fails.                                     03462
      This procedure will first scan the list for the number of items
      desired.  For help, the number is one and the search will stop
      after the first item is found.  For query, this may be "all" or
      a bounded set as well.  If it is not the default of one, this
      procedure finds the multiple occurances and builds up the
      context stack (in this case a multiple occurance stack).  If it
      is one, this procedure finds the single item, but merely
      initializes the menu stack which will be built up by the print
      routine as substructure is discovered. %                      03463
      LOCAL                                                         03464
         conad, %address of current context%                        03465
         stid, % of found item or -1 %                              03466
         nwindex, % of ring or old index if not found %             03467
         ended, % TRUE when ENDCHR reached %                        03468
         firstflg, %set for first item on list%                     03469
         numflg, %set when list item is a menu number%              03470
         upflg,                                                     03471
         wkstd,                                                     03472
         retval, % RETURN flag - TRUE if something new to be printed,
         FALSE if there is nothing new to print%                    03473
```

```
   number,                                                          03474
   qswork[7]; % work area for string searches in scanit %           03475
LOCAL STRING testr[50]; % String to be used for each item in
the list %                                                          03476
REF list, conad;                                                    03477
% Initialize for search %                                          03478
   retval ← FALSE;                                                  03479
   stid ← -1;                                                       03480
   ended ← FALSE;                                                   03481
   nwindex ← index;                                                 03482
   &conad ←                                                         03483
      IF index = entind THEN $entcon                                03484
      ELSE &conrng + index*rnglnt;                                  03485
   firstflg ← TRUE;                                                 03486
   IF NOT list.L THEN RETURN(FALSE, stid, index);                   03487
   % Set up scan area. %                                            03488
      qswork ← &list;                                               03489
      qswork.stastr ← 1;                                            03490
      qswork[1] ← 1;                                                03491
      fechcl(forward, $qswork);                                     03492
% Scan for back characters at the beginning of the list %          03493
   CASE READC($qswork) OF                                           03494
      ='<:                                                          03495
         BEGIN                                                      03496
         % Backup a context %                                       03497
         retval ← TRUE; % There is always something to print
         after a back %                                             03498
         IF nwindex # entind THEN                                   03499
            BEGIN                                                   03500
            nwindex ← conbck( nwindex );                            03501
            &conad ← IF nwindex = entind THEN $entcon               03502
               ELSE &conrng + nwindex*rnglnt;                       03503
            stid ← conad;                                           03504
            END;                                                    03505
         REPEAT CASE;                                               03506
         END;                                                       03507
      =',, = NP: REPEAT CASE;                                       03508
      =ENDCHR:                                                      03509
         RETURN(IF stid # endfil THEN retval ELSE FALSE, stid,
         nwindex);                                                  03510
      ENDCASE %backup pointer%                                      03511
         IF qswork[1] # 1 THEN                                      03512
            BEGIN                                                   03513
            qswork[1] ← qswork[1]-1;                                03514
            fechcl(forward,  $qswork);                              03515
            END;                                                    03516
%scan for list items and search for each %                         03517
   UNTIL ended DO % over list until it is exhausted %               03518
      BEGIN                                                         03519
      % Scan for the next item in the list pointed to by the
      byte pointer bfrom %                                          03520
         IF NOT scanit( $testr, $qswork : ended, upflg, numflg)
         THEN                                                       03521
            BEGIN                                                   03522
            dismes(2, $"Invalid List Characters");                  03523
            IF firstflg THEN RETURN (retval,stid,nwindex);
```

```
                                                                    03524
      RETURN(retval,stid,nwindex);                                  03525
      END;                                                          03526
   IF upflg THEN                                                    03527
      BEGIN                                                         03528
      IF stid = endfil THEN stid ← conad;                           03529
      IF (wkstd ← getup(stid)).stpsid # orgstid THEN
                                                                    03530
         stid ← wkstd;                                              03531
      retval ← TRUE;                                                03532
      firstflg ← FALSE;                                             03533
      END;                                                          03534
   IF numflg = -1 OR NOT testr.L THEN REPEAT LOOP;                  03535
   IF numflg THEN number ← VALUE($testr);                           03536
% Searcher%                                                        03537
CASE TRUE OF                                                        03538
   = firstflg .A numflg:% Search A:  first item in the
   list is a menu number %                                          03539
      BEGIN                                                         03540
      IF NOT numfst(&conad, stid, number, nwindex: stid,
      nwindex) THEN                                                 03541
         RETURN(retval,stid, nwindex);                              03542
      retval ← TRUE; % We have successfully found the
      first list item so there will always be something
      to print. %                                                   03543
      firstflg ← FALSE;                                             03544
      END;                                                          03545
   = numflg:% Search B:  list item is a number but not
   the first item %                                                 03546
      IF NOT qnum(stid,number:stid) THEN                            03547
         RETURN(retval,stid, nwindex);                              03548
   = firstflg:% Search D:  Name is first item;  Search
   all context stacks and sequentially from the top.%
                                                                    03549
      BEGIN                                                         03550
      IF NOT namfst($testr,&conad,stid,
      nwindex:stid,nwindex) THEN                                    03551
         BEGIN                                                      03552
         IF NOT topsch($testr:stid) THEN                            03553
            BEGIN                                                   03554
            %Check for more files in the data base.  If
            more search them sequentially%                          03555
            RETURN(retval,stid,index);                              03556
            END;                                                    03557
         END;                                                       03558
      firstflg ← FALSE;                                             03559
      retval ← TRUE;                                                03560
      END;                                                          03561
   ENDCASE % not firstflg and not numflg.  Do a
   sequential search under he current location and give
   up.%                                                             03562
      BEGIN                                                         03563
      IF NOT seqsrch($testr,stid: stid) THEN                        03564
         RETURN (retval,stid,nwindex);                              03565
      END;                                                          03566
END;                                                                03567
```

```
      RETURN(retval,stid, nwindex);                                    03568
   END.
                                                                       03569
(scanit) PROCEDURE (testr, qswork);                                    03570
   LOCAL numflg, char, upflg;                                          03571
   REF testr, qswork;                                                  03572
   % Scanner initialization %                                          03573
      testr.L ← 0;                                                     03574
      numflg ← -1; % TRUE (1) is number, FALSE (0) is alpha, -1 is     
      neither. %                                                       03575
      upflg ← FALSE;                                                   03576
   % get next item in list into testr for checking %                   03577
      CASE char ← READC(&qswork) OF                                    03578
         = L:                                                          03579
            BEGIN                                                      03580
            *testr* ← *testr*, char;                                   03581
            numflg ← FALSE;                                            03582
            CASE char ← READC(&qswork) OF                              03583
               = LD, ='-, ='', ='@:                                    03584
                  BEGIN                                                03585
                  *testr* ← *testr*, char;                             03586
                  REPEAT CASE;                                         03587
                  END;                                                 03588
               =',:                                                    03589
                  BEGIN                                                03590
                  RETURN( TRUE, % scan was OK %                        03591
                     FALSE, % list is not ended %                      03592
                     upflg, numflg);                                   03593
                  END;                                                 03594
               =NP:                                                    03595
                  CASE READC(&qswork) OF                               03596
                     =NP: REPEAT CASE;                                 03597
                     =ENDCHR:                                          03598
                        RETURN( TRUE,                                  03599
                           TRUE,                                       03600
                           upflg, numflg);                             03601
                     ENDCASE                                           03602
                        BEGIN                                          03603
                        IF qswork[1] # 1 THEN                          03604
                           BEGIN                                       03605
                           qswork[1] ← qswork[1]-1;                    03606
                           fechcl(forward,  &qswork);                  03607
                           END;                                        03608
                        RETURN(TRUE,                                   03609
                           FALSE,                                      03610
                           upflg, numflg);                             03611
                        END;                                           03612
               = ENDCHR:                                               03613
                  BEGIN                                                03614
                  RETURN( TRUE, % scan was OK %                        03615
                     TRUE, % list is ended %                           03616
                     upflg, numflg);                                   03617
                  END;                                                 03618
               ENDCASE % Illegal character in the name%                03619
                  BEGIN                                                03620
                  % Type out error message, reset return
```

```
                    variables, and exit for return %                    03621
                    RETURN( FALSE, % scan was not OK %                  03622
                       FALSE, % list is ended %                         03623
                       upflg, numflg);                                  03624
                    END;                                                03625
                END;                                                    03626
             = D:                                                       03627
                BEGIN                                                   03628
                *testr* ← *testr*, char;                                03629
                numflg ← TRUE;                                          03630
                CASE char ← READC(&qswork) OF                           03631
                   = D:                                                 03632
                      BEGIN                                             03633
                      *testr* ← *testr*, char;                          03634
                      REPEAT CASE;                                      03635
                      END;                                              03636
                   =',:                                                 03637
                      BEGIN                                             03638
                      RETURN( TRUE, % scan was OK %                     03639
                         FALSE, % list not ended %                      03640
                         upflg, numflg);                                03641
                      END;                                              03642
                   =NP:                                                 03643
                      CASE READC OF                                     03644
                         =NP: REPEAT CASE;                              03645
                         =ENDCHR:                                       03646
                            RETURN( TRUE,                               03647
                               TRUE,                                    03648
                               upflg, numflg);                          03649
                         ENDCASE                                        03650
                            BEGIN                                       03651
                            IF qswork[1] # 1 THEN                       03652
                               BEGIN                                    03653
                               qswork[1] ← qswork[1]-1;                 03654
                               fechc1(forward,  &qswork);               03655
                               END;                                     03656
                            RETURN(TRUE,                                03657
                               FALSE,                                   03658
                               upflg, numflg);                          03659
                            END;                                        03660
                   = ENDCHR:                                            03661
                      BEGIN                                             03662
                      RETURN( TRUE, % scan was OK %                     03663
                         TRUE, % list is ended %                        03664
                         upflg, numflg);                                03665
                      END;                                              03666
                   ENDCASE % Illegal character in the number%           03667
                      BEGIN                                             03668
                      % Type out error message, reset variables , and
                      exit for return %                                 03669
                      RETURN( FALSE, % scan was not OK %                03670
                         FALSE, % list is not ended %                   03671
                         upflg, numflg);                                03672
                      END;                                              03673
                END;                                                    03674
             = '↑:                                                      03675
```

```
            BEGIN                                                   03676
            testr.L ← 0;                                            03677
            upflg ← TRUE;                                           03678
            REPEAT CASE;                                            03679
            END;                                                    03680
         = ',, =NP:                                                 03681
            RETURN( TRUE, % scan was OK %                           03682
               FALSE, % list is not ended %                         03683
               upflg, numflg);                                      03684
         = ENDCHR:                                                  03685
            RETURN( TRUE, % scan was OK %                           03686
               TRUE, % list is ended %                              03687
               upflg, numflg);                                      03688
         ENDCASE                                                    03689
            BEGIN                                                   03690
            % Type out error message, reset variables , and exit
            for return %                                            03691
            RETURN( FALSE, % scan was not OK %                      03692
               FALSE, % list is not ended %                         03693
               upflg, numflg);                                      03694
            END;                                                    03695
   END.
                                                                    03696
(conbck) PROCEDURE ( index ); % back up one context if possible.
otherwise go to etrypoint %                                         03697
   % Check for entrypoint BEFORE callng this procedure!!! %        03698
   CASE index OF                                                    03699
      >0: IF (index ← index-1) = confre THEN index ← entind;        03700
      ENDCASE %=0%                                                  03701
         index ← IF conrng[rngmax*2] THEN rngmax ELSE entind;
                                                                    03702
   RETURN( index );                                                 03703
   END.
                                                                    03704
(qvalid) PROCEDURE ( conad, index ); % Check if this stack has any
entries; back up if not %                                           03705
   REF conad;                                                       03706
   UNTIL conad.constk OR index = entind DO                          03707
      BEGIN                                                         03708
      index ← conbck( index );                                      03709
      &conad ←                                                      03710
         IF index   entind THEN $entcon                             03711
         ELSE &conrng + index*rnglnt;                               03712
      END;                                                          03713
   RETURN(&conad, index );                                          03714
   END.
                                                                    03715
(numfst)PROCEDURE (conad, stid, number, index); % The first item
in the list is a menu number.  Count down through menu stack "n"
items.  It is an error if the number is too high%                   03716
   LOCAL wrkaddr, nwindex;                                          03717
   LOCAL STRING message[100];                                       03718
   REF conad, wrkaddr;                                              03719
   &conad ← qvalid(&conad, index: nwindex);                         03720
   IF conad.totcnt < number THEN                                    03721
      BEGIN                                                         03722
```

```
      *message* ← STRING(number)," is an invalid menu number";
                                                                    03723
      dismes(2,$message); % for the display, we should perhaps put
      message on the screen%                                        03724
      RETURN (FALSE,stid, index);                                   03725
      END;                                                          03726
   &wrkaddr ← conad.constk;                                         03727
   LOOP %over menu blocks in this stack %                           03728
      BEGIN                                                         03729
      IF number < stkmax THEN                                       03730
         BEGIN                                                      03731
         stid ← wrkaddr[number*2];                                  03732
         RETURN (TRUE,stid, nwindex);                               03733
         END                                                        03734
      ELSE                                                          03735
         BEGIN                                                      03736
         &wrkaddr ← wrkaddr;                                        03737
         number ← number - stkmax;                                  03738
         END;                                                       03739
      END;                                                          03740
   END.
                                                                    03741
(qnum)PROCEDURE(stid,number); %A menu number found under a
superior list item.  Scan first level substructure under the
current stid for n-1 successors.  If invalid (i.e., getsuc
encounters an 'up' before it reaches n-1 give error message,
return TRUE or FALSE (if number invalid) and "current " location.%
                                                                    03742
   LOCAL wrkstd,count,savstd;                                       03743
   LOCAL STRING message[100];                                       03744
   count ← number;                                                  03745
   IF (wrkstd ← getsub(stid)) = stid OR                             03746
      (count ← count-1) < 0 THEN                                    03747
      BEGIN                                                         03748
      *message* ← STRING(number)," is an invalid menu number";
                                                                    03749
      dismes(2, $message);                                          03750
      RETURN (FALSE,stid);                                          03751
      END;                                                          03752
   FOR count DOWN UNTIL <= 0 DO                                     03753
      BEGIN                                                         03754
      IF (savstd ← getsuc(wrkstd)) = stid THEN                      03755
         BEGIN                                                      03756
         %no successors so return error  condition%                 03757
         *message* ← STRING(number)," is an invalid number";        03758
         dismes(2,$message);                                        03759
         RETURN (FALSE,stid);                                       03760
         END                                                        03761
      ELSE wrkstd ← savstd;                                         03762
      END;                                                          03763
   RETURN(TRUE,wrkstd);                                             03764
   END.
                                                                    03765
(namfst)PROCEDURE(testr, conad, stid, index);                       03766
   %a name is first list item.  Look through menu for one context
   stack.  Look through all context stckss.  Look through data
```

```
    base sequentially from the top.%                                    03767
    LOCAL nhash,levcnt,stccnt,wrkaddr,blkcnt, wrkstd,wkindex;           03768
    LOCAL STRING locstr[30];                                            03769
    REF conad,testr,wrkaddr;                                            03770
    levcnt ← 4;                                                         03771
    IF topsch(&testr: stid) THEN RETURN(TRUE,stid,index)                03772
    ELSE RETURN(FALSE,stid,index);                                      03773
    wkindex ← index;                                                    03774
    astruc(&testr); %convert to upper case%                             03775
    nhash ← hash(&testr); %hash name %                                  03776
    blkcnt ← 0;                                                         03777
    &conad ← qvalid(&conad, wkindex: wkindex);                          03778
    &wrkaddr ← conad.constk;                                            03779
    LOOP                                                                03780
       BEGIN                                                            03781
       DO                                                               03782
          BEGIN                                                         03783
          UNTIL (stccnt ← stccnt + 1) > wrkaddr.stkcnt DO               03784
             IF nhash = wrkaddr[stccnt*2+1] THEN                        03785
                BEGIN % found a match on name hash %                    03786
                wrkstd ←wrkaddr[stccnt*2];                              03787
                CCPOS SF(wrkstd);                                       03788
                xtrnam($locstr,$swork,-1);                              03789
                IF *testr* = *locstr* THEN                              03790
                   RETURN (TRUE,wrkaddr[stccnt*2],wkindex);             03791
                END;                                                    03792
          END                                                           03793
       UNTIL (blkcnt ← blkcnt + stkmax) > conad.totcnt;                 03794
       % Switch contexts if appropriate.  We exhausted all blocks
       in this context with no match.%                                  03795
          RETURN (FALSE, stid, index);                                  03796
       END;                                                             03797
    END.
                                                                        03798
(seqsrch) PROCEDURE (testr,stid); %Do a name-in-group search
sequentially under the current location - if this is the top of a
file the whole file will be searched%                                   03799
    LOCAL wrkstd;                                                       03800
    IF (wrkstd ← namingrp(stid,stid,testr,1000)) = endfil THEN
                                                                        03801
       BEGIN                                                            03802
       RETURN (FALSE,stid);                                             03803
       END                                                              03804
    ELSE RETURN (TRUE,wrkstd);                                          03805
    END.
                                                                        03806
(topsch) PROCEDURE(testr); %Set  current location to top of data
base. Call seqsrch to search the file.  Check to see if there are
more files in the data base.%                                           03807
    LOCAL wrkstd;                                                       03808
    IF NOT multiflg THEN                                                03809
       BEGIN                                                            03810
       IF NOT seqsrch(testr,topcon: wrkstd) THEN                        03811
          RETURN(FALSE, endfil)                                         03812
       ELSE RETURN(TRUE,wrkstd);                                        03813
       END;                                                             03814
```

```
    % multiple file data base %                                    03815
    % Access the directory table and get the next file after closng
    the last file.  Search each file (seqsrch) until end of
    directory is reached.%                                         03816
    RETURN(FALSE, endfil);                                         03817
    END.
                                                                   03818
(neighb)PROCEDURE(testr, stid, levcnt); % Neighbor search for a
    name.  The search will end when the number of levels down
    specified by levcnt from stid has been searched.  If stid is the
    origin the whole file (down to levcnt) will be searched.  If tid
    is a statement other than the orgin the search will be confined to
    that branch. %                                                 03819
       LOCAL                                                       03820
          wkstid,                                                  03821
          curlev, % the current level being searched %             03822
          nhash; % has of searched for name %                      03823
       IF stid = endfil THEN RETURN(FALSE, stid); % should never
       happen! %                                                   03824
       astruc(testr); %convert tc upper case %                     03825
       nhash ← hash(testr); % hash it %                            03826
```

Similar to NEWNAME

```
                                                                    03828
      RETURN(TRUE, wkstid);                                         03829
   % Check the rest %                                               03830
      FOR curlev ← 1 UP UNTIL > levcnt DO                           03831
         BEGIN                                                      03832
         IF (wkstid ← levnam(stid, testr, nhash, curlev)) # endfil
                                                                    03833
         THEN                                                       03834
            RETURN(TRUE, wkstid);                                   03835
         END;                                                       03836
   % Get here if search failed on all levels in this file. %       03837
      RETURN(FALSE, stid);
   END.
                                                                    03838
(levnam) %lookup name in file starting at stid and terminating
when through with its branch headed by lbstid, looking down levels
levels checking names only on the last level.%                      03839
   % This procedure finds the statement whose name is in namstr.
   the search is restricted to that part of the file begining with
   STID through its branch, and within LEVELS levels below STID. %
                                                                    03840
   %--------------------%                                          03841
   PROC(stid, namstr, nhash, levels);                               03842
   LOCAL save, lbstid;                                              03843
   LOCAL STRING locstr[100];                                        03844
   REF namstr;                                                      03845
   save ← rubabt := FALSE;                                          03846
   % Set up terminating stid %                                      03847
      lbstid ← stid;                                                03848
   LOOP                                                             03849
      BEGIN                                                         03850
      IF NOT levels AND getnmf(stid) AND getnam(stid) = nhash THEN
                                                                    03851
         BEGIN                                                      03852
```

```
            CCPOS SF(stid);                                              03853
            xtrnam($locstr, $swork, -1);                                 03854
            IF *namstr* = *locstr* THEN                                  03855
               BEGIN                                                     03856
               rubabt ← save;                                            03857
               RETURN(stid);                                             03858
               END;                                                      03859
            END;                                                         03860
         IF levels > 0 AND (stid := getsub(stid)) # stid THEN            03861
            BUMP DOWN levels                                             03862
         ELSE                                                            03863
            LOOP                                                         03864
               IF getftl(stid) THEN                                      03865
                  BEGIN                                                  03866
                  BUMP levels;                                           03867
                  IF stid = lbstid OR inptrf THEN EXIT LOOP 2;           03868
                  stid ← getsuc(stid); %GETSUC RETURNS THE UP IF
                  TAIL%                                                  03869
                  END                                                    03870
               ELSE                                                      03871
                  BEGIN                                                  03872
                  IF stid = lbstid OR inptrf OR stid.stpsid = orgstid
                  THEN                                                   03873
                     EXIT LOOP 2;                                        03874
                  stid ← getsuc(stid);                                   03875
                  EXIT LOOP;                                             03876
                  END;                                                   03877
            END;                                                         03878
      rubabt ← save;                                                     03879
      RETURN(endfil);                                                    03880
      END.
                                                                         03881
                                                                         03882
% Sequence generator(s) for printg %                                    03883
   (queryseq) PROC (sw, phase); %query sequence generator program%
                                                                         03884
      %This controls the printout or display of a QUERY branch when a
      SHOW function is requested.  It handles INCLUDES, Q-specs, and
      menu generation.%
                                                                         03885
      LOCAL saveview;                                                    03886
      REF sw;
                                                                         03887
      CASE phase OF                                                      03888
         = sqopn: %OPEN%                                                 03889
            RETURN;                                                      03890
         = sqgnxt: %NEXT (below)%                                        03891
            NULL;                                                        03892
         = sqcls: %CLOSE%                                                03893
            RETURN;                                                      03894
         ENDCASE abortsubsystem($"Query System Failure");
                                                                         03895
      ON SIGNAL ELSE                                                     03896
         BEGIN                                                           03897
         % Some awful failure; send back EOF with error flag set. %
                                                                         03898
```

```
      hseqer ← TRUE; % Error message will be wrong, but... % 03899
      sw.swcstid ← sw.swstid ← endfil;                      03900
      sport(&sw); %don't want to be called again.  Finished with a
      query branch%                                         03901
      END;                                                  03902
   qnewstmt.L ← 0;                                          03903
   saveview ← sw.swvspec;                                   03904
   qstmt(&sw, 0, %no previous sequence% TRUE, % do process the
   subplex of this statement, if any % $""); %process addressed
   node top statement and build a menu for any substructure% 03905
   sw.swvspec ← saveview;                                   03906
   % send back EOF because we are done %                    03907
      sw.swcstid ← sw.swstid ← endfil;                      03908
      sport(&sw); %don't want to be called again.  Finished with a
      query branch%                                         03909
   END.
                                                            03910
(qstmt) PROC(sw, prevsw, topflag, appendstr); %process one QUERY
source statement%                                           03911
   % Examine "current" data base source statement, executing any
   links and performing INCLUDES wherever context and Q-specs
   dictate.  Trigger menu generation only if topflag=TRUE.  ELSE
   ignore substructure to this statement.%                  03912
   % If topflag = TRUE, assumed to be top addressed node, i.e. an
   asterisk as the first character must be stripped off %    03913
   % qstmt will either be called                            03914
      1.  by queryseq to process the branch being SHOWN
      (addressed) (prevsw=0)                                03915
      2.  by qmenu to process a single statement in the plex being
      menued (uncolumnated only) (prevsw#0) %
                                                            03916
   LOCAL                                                    03917
      code,   % Type of statement: menu, non-menu, etc. %   03918
      truncate,  % Do not execute links beyond the frist line if
      TRUE. %                                               03919
      sendsw,   % temp for sequence work area address%      03920
      nextchr,  % count of next character to be scanned. %  03921
      dirflg;% TRUE = q-directive found %
                                                            03922
   LOCAL TEXT POINTER                                       03923
      oldpos,  % front of this segment %                    03924
      endnew;   % end of this segment %                     03925
   REF sw, prevsw, sendsw, appendstr; %could be top node or a
   linked-to INCLUDED node%
                                                            03926
   %general initialization%                                 03927
      oldpos ← endnew ← sw.swcstid;                         03928
      endnew[1] ← oldpos[1] ← CASE code ← tstmenu(oldpos) OF 03929
         = 3, = 4: 2; % Comment character to be peeled off % 03930
         ENDCASE 1; % Logical front of this statement %     03931
      truncate ← NOT topflag AND code IN [1,2];             03932
      &sendsw ← IF &prevsw THEN &prevsw ELSE &sw;
                                                            03933
   %Handle contents of this statement.  Optimize case where no
   directives occur.  No FIND statements used until queryspecs
   encountered.%                                            03934
```

```
      LOOP                                                          03935
         BEGIN % keep processing through this stmt.  If this is a
         top level node, process  until segments of text are sent
         and all link list INCLUDES have been taken.  If this is a
         menu item, gather data in global string for single send
         when includes have been taken; avoid CRs generated by
         sends. %                                                   03936
         dirflg ← qdirparse ($oldpos, $endnew : nextchr);           03937
         % append and send previous segment if this is top level
         node; otherwise, just append %                             03938
            dirflg ← qappender($qnewstmt, &appendstr, $oldpos,
            $endnew, truncate, dirflg, sendsw.swvspec.vstrnc);
                                                                    03939
               % If this is a menu node, only append first 72
               characters or one line; change dirflg to reflect
               limit reached to exit loop. %                        03940
            IF topflag OR NOT dirflg THEN % send out now. %         03941
               qsender($qnewstmt, &sendsw, topflag);                03942
            IF NOT dirflg THEN EXIT LOOP;                           03943
         % advance pointers for include %                            03944
            oldpos[1] ← endnew[1] := nextchr;                       03945
         % do link list found by qdirparse - whether called with a
         menu statement or a main node %                            03946
            qinclude (&sendsw, $qnewstmt, $oldpos, $endnew,
            topflag);                                               03947
               % qinclude calls qstrip to process included stmt,
               Qstrip sends the stmt out without directives %
                                                                    03948
         % advance pointers for next scan %                          03949
            oldpos[1] ← endnew[1] ← nextchr;                        03950
         END; % of text/include loop %                              03951
      IF topflag THEN % we are now processing the TOP statement
      (called from queryseq) and about to do MAIN subplex %         03952
         qmenu (sw.swcstid, sw.swlbstid, &sendsw); %do menu as
         requested%                                                 03953
   RETURN; END.
    %normal exit%
                                                                    03954
(qsender) PROC (string, workarea, topflag);                         03955
   % sends the contents of string out to sequence workarea with
   level controlled by topflag.  Resets contents of string after
   send. %                                                          03956
   %It avoids sending a null string%
                                                                    03957
   LOCAL savlvl;                                                    03958
   REF string, workarea;
                                                                    03959
   IF string.L THEN                                                 03960
      BEGIN                                                         03961
      savlvl ← workarea.swclvl := workarea.swslvl + (NOT topflag);
                                                                    03962
      send(&workarea, &string);                                     03963
      workarea.swclvl ← savlvl;                                     03964
      string.L ← 0;                                                 03965
      END;                                                          03966
   RETURN END.
```

```
                                                                    03967
(qappender) PROC (string, appgndstr, front, end, truncate, dirflg,
lines);                                                             03968
   % Appends contents of appendstr to front of string; puts
   contents delimited by front and end at the end of string.
   (Does not try to append sppendstr if NULL; does not try to
   append front through end if front is zero.)  Does not reset
   original contents of the string. %                               03969
   LOCAL length;                                                    03970
   LOCAL TEXT POINTER tp;                                           03971
   REF string, appendstr, front, end;
                                                                    03972
   IF appendstr.L THEN                                              03973
      BEGIN                                                         03974
      *string* ← *appendstr*, *string*;                             03975
      appendstr.L ← 0;                                              03976
      END;                                                          03977
   IF front THEN *string* ← *string*, front end;                    03978
   IF truncate  THEN                                                03979
      BEGIN                                                         03980
      IF (length ← lines*72) < string.L THEN                        03981
         BEGIN                                                      03982
         string.L ← length;                                         03983
         dirflg ← FALSE;                                            03984
         END;                                                       03985
      IF FIND SF(*string*) [EOL] ↑tp ←tp THEN                       03986
         BEGIN                                                      03987
         *string*  ← SF(*string*) tp;                               03988
         dirflg ← FALSE;                                            03989
         END;                                                       03990
      END;                                                          03991
   RETURN(dirflg);                                                  03992
   END.
                                                                    03993
(qdirparse) PROC (start, end); %parse query directives%             03994
   % Given the address of a text pointer to the current position
   in the statement (which will NOT be changed by this procedure),
   the address of a text pointer which may be changed to point to
   the beginning of a qspec block or the end of the statement if
   no block exists, qdirparse locates the query directive string,
   if any, sets end, and returns TRUE and the count character of
   the next character to be scanned.                                03995
   If no directives are found, it returns FALSE and the count of
   the end of the statement.                                        03996
   It attempts to do the scan quickly to optimize case where no
   directives exist. %
                                                                    03997
   LOCAL                                                            03998
      count,                                                        03999
      drswork[7]; % string work area for READC. %                   04000
   REF start, end;                                                  04001
   % Initialize the string work area. %                            04002
      drswork ← start;                                              04003
      drswork[1] ← start[1];                                        04004
      fechcl(forward, $drswork);                                    04005
      count ← 0;                                                    04006
```

```
    % Find the first. %                                                  04007
       CASE READC($drswork) OF                                          04008
          = '#:                                                         04009
             BEGIN                                                      04010
             IF NOT count THEN                                          04011
                BEGIN                                                   04012
                IF READC($drswork) # '# THEN REPEAT CASE;               04013
                end[1] ← drswork[1] - 2;                                04014
                BUMP count;                                             04015
                REPEAT CASE;                                            04016
                END                                                     04017
             ELSE                                                       04018
                BEGIN                                                   04019
                IF READC($drswork) # '# THEN REPEAT CASE;               04020
                RETURN( TRUE, drswork[1]);                              04021
                END;                                                    04022
             END;                                                       04023
          =ENDCHR:                                                      04024
             BEGIN                                                      04025
             RETURN( FALSE, end[1] ← drswork[1]);                       04026
             END;                                                       04027
          ENDCASE REPEAT CASE;                                          04028
    END.                                                                04029

(qinclude) PROC (sw, string, qlkleft, qlkright, topflag); %process
query link list (includes)%                                             04030
       %Process from global qlkleft to qlkright, obeying user's
       viewspecs in links and the qspecs directives.  Each link in the
       list is executed, and a new sequence is opened at the new
       location. User viewspecs, if any, predominate over query
       defaults.                                                        04031
       The linked-to statement is included without its statement name
       and without any directives ("##....##") it may contain.  Qspecs
       govern the inclusion of the substructure as menu.                04032
       Before leaving, the file and new sequence are closed.            04033
          File not closed for now; need file management code to take
          care of stids if files must be closed.                        04034
       PARAMETERS:  address of sequence work area (main), flag for
       including name on linked-to statement(s).  If topflag is true,
       we are processig the topnode or links in the top node.  This
       affects whether or not we will accept new qspecs.  Generally no
       if false-- formatting based on the top node. %                   04035

       %NOTE:  assuming format of "##[..](..)(..)[..](..)etc##  If a
       queryspecs setting occurs at the end of the string, it will
       pertain to the substructure of this node (if applicable) since
       it stands alone.  This, of course, will be erased if another
       list occurs before substructure of this node processed.  To set
       qspecs to affect the sub of the node begin addressed, use a
       [..] last in the list (or only element).%                        04036

       LOCAL                                                            04037
          nofind,   % to avoid looping: TRUE if neither item located %  04038

          stid;     % for new file/stmt %                               04039
       LOCAL TEXT POINTER                                               04040
```

```
   curpos,     % current position in string being processed %
   oldpos,     % temp for position %
   startspec,    % start of qspec or link %
   endspec;      % end of qspec or link %
LOCAL STRING qstn[200]; %statement number%
REF
   sw,
   string,
   qlkleft,
   qlkright;

ON SIGNAL ELSE RETURN; %currently, this is a way out - lnkspec
will complain when it finds no more links%

% initialization %
   IF qlkleft[1] = qlkright[1] THEN RETURN;
   stid ← curpos ← oldpos ← qlkleft;  % in case no link, just
   viewspecs %
   curpos[1] ← endspec[1] ← qlkleft[1];
   % strip garbage off end of link list %
      FIND qlkright < 2CH $NP ↑qlkright;
% run through list, getting [..] first, then (...)(...) etc. %

LOOP
   BEGIN
   % process QSPECS   %
      nofind ← FALSE;
      IF FIND curpos > 2CH $NP ↑endspec '/ ↑startspec  $(LD/'=)
       %NOTE:  really ought to use FS and let anything go by,
      just trapping ) for this to handle errors in the best
      way% ') < CH ↑endspec THEN
         % found QSPECS %
         BEGIN
         qsparse ($startspec, $endspec, topflag);
         curpos[1] ← endspec[1] + 1; % next char to process %

         END
      ELSE BUMP nofind;
      IF (oldpos[1] ← curpos[1]) >= qlkright[1] THEN EXIT LOOP;

   % process next link %
      % parse link and get stid. %
         qlnkspec($curpos, $stno, $stn2 %filename%,$qstn
         %number%, $num %vspecs%); % this will go to err if no
         more links %
         IF stn2.L % filename exists % THEN
            stid ← nfstid($stn2 %filename%, $qstn %stmt no%,
            $num %vspecs%) %open the file%
         ELSE stid ← ofstid( &qda, $qstn);
      %concatenate user string with ours (esb), ours first;
      change viewspecs only if not being menued.  Otherwise
      must only put out one line!%
         IF topflag THEN *num* ← "esb", *num*
         ELSE *num* ← "te"; % one line, one level in menu %
```

```
            feedsw(&sw, $num %vspecs%); %put new viewspecs in the
            sequence work area%                                    04080
         %remove directives, send out%                             04081
            qstrip(stid, &sw % destination of qstrip's output %,
            topflag, &string);                                     04082
         IF topflag AND qspecs.querinc THEN % will not be TRUE if
         NOT topflag (see qsparse) %                               04083
            BEGIN                                                  04084
            qmenu(stid, endfil, &sw); %do a menu if
            user-specified%                                        04085
            END;                                                   04086
      % test for done %                                           04087
         CASE curpos[1] OF                                         04088
            = oldpos[1]: % no more links in statement %            04089
               EXIT LOOP;                                          04090
            >= qlkright[1]: % end of section of stmt to process %
                                                                   04091
               EXIT LOOP;                                          04092
            ENDCASE nofind ← FALSE; % we found link %              04093
         BUMP curpos[1]; % advance to next character %             04094
      IF nofind THEN EXIT; % didn't find [...] or (...) - DB
      error.  Just ignore %                                        04095
      END; % of qspecs/links processing loop %                     04096
   RETURN END.
                                                                   04097
(qsparse) PROC(qspstrt, end, topflag); %set global queryspecs%
                                                                   04098
   %interpret string between qspstrt and end and set proper flags
   in global, qspecs - WARNING to data base builder - this is a
   hard-nosed son-of-a-bitch routine.%
                                                                   04099
   LOCAL                                                           04100
      char,    % next char to process %                           04101
      number,                                                      04102
      term, % number of characters to scan %                       04103
      qspwork[7]; %string work area%                               04104
   LOCAL STRING nmstr[10];                                         04105
   REF qspstrt, end;
                                                                   04106
   % initialize string work area. %                               04107
      qspwork ← qspstrt;                                           04108
      qspwork[1] ← qspstrt[1];                                     04109
      fechcl( forward,  $qspwork);                                 04110
      term ← end[1] - qspstrt[1];                                  04111
   number ← FALSE;                                                 04112
   qspcreset(topflag); %set default values for qspecs%             04113
   FOR term DOWN UNTIL <= 0 DO                                     04114
      BEGIN                                                        04115
      CASE char ← READC($qspwork) OF %get next char of statement%
                                                                   04116
         ='c, ='C:                                                 04117
            IF topflag THEN qspecs.quercol ← TRUE;                 04118
         ='m, ='M: % Do not include substructure.-- Main node only
         %                                                         04119
            IF topflag THEN qspecs.querinc ← FALSE;                04120
         ='p, ='P:                                                 04121
```

```
            qspecs.querprt ← TRUE;                                  04122
         ='n, ='N:                                                  04123
            number ← TRUE;                                          04124
         ='=:                                                       04125
            IF NOT number:=number+1 THEN EXIT LOOP;                 04126
         = D: %digit%                                               04127
            BEGIN                                                   04128
            IF number # 2 THEN EXIT LOOP;                           04129
            IF nmstr.L = nmstr.M THEN EXIT LOOP; %avoid overflow%
                                                                    04130
            *nmstr* ← *nmstr*, char;                                04131
            END;                                                    04132
         = NP: REPEAT LOOP;                                         04133
         ENDCASE EXIT LOOP;                                         04134
      END; %LOOP%                                                   04135
   IF topflag AND nmstr.L THEN                                      04136
      qspecs.queropts ← VALUE($nmstr); %may truncate value!%        04137
   RETURN END.
                                                                    04138
(qspcreset) PROC (topflag); %reset default queryspecs values%
                                                                    04139
   %qspecs is global%                                               04140
   IF topflag THEN                                                  04141
      BEGIN                                                         04142
      qspecs.quercol ← FALSE;                                       04143
      qspecs.querinc ← TRUE;                                        04144
      qspecs.queropts ← FALSE;                                      04145
      END;                                                          04146
   qspecs.querprt ← FALSE;                                          04147
   RETURN END.
                                                                    04148
(qstrip) PROC (stid, destsw, topflag, string); % decrud a data
base stmt %                                                         04149
   % THIS IS ALMOST IDENTICAL TO QSTMT!!! DROP ONE. %              04150
   % Given the source sequence work area address, and the
   destination sequence work area address, drop off special query
   commands  and port-send the string out%
                                                                    04151
   LOCAL                                                            04152
      code,   % Type of statement: menu, non-menu, etc. %           04153
      truncate,  % Do not execute links beyond the frist line if
      TRUE. %                                                       04154
      dirflg,    % TRUE if directive found %                       04155
      nextchr;   % curent next char to process %                   04156
   LOCAL TEXT POINTER                                               04157
      oldpos,    % front of text to SEND %                         04158
      endnew;      % end of text to SEND %                         04159
   REF destsw, string;
                                                                    04160
   % initialize %                                                   04161
      oldpos ← endnew ←  stid;                                      04162
      oldpos[1] ← endnew[1] ←                                       04163
         CASE code ← tstmenu(stid) OF                               04164
            = 3, = 4: 2; % Comment character to be peeled off %
                                                                    04165
            ENDCASE 1; % Logical front of this statement %          04166
```

```
      truncate ← NOT topflag AND code IN [1,2];                    04167
   %Handle contents of this statement.  Optimize case where no
   directives occur.  NO FIND statements used until queryspecs
   encountered.%                                                   04168
      LOOP                                                        04169
         BEGIN % keep processing through this stmt.  If this is a
         top level node, process  until segments of text are sent.
          If this is a menu item, gather data in global string for
         single send when includes have been taken; avoid CRs
         generated by sends. %                                    04170
         dirflg ← qdirparse ($oldpos, $endnew : nextchr);         04171
         % append and send previous segment if this is top level
         node; otherwise, just append %                           04172
            dirflg ← qappender(&string, $"", $oldpos, $endnew,
            truncate, dirflg, destsw.swvspec.vstrnc);             04173
               % If this is a menu node, only append first 72
               characters or one line; change dirflg to reflect
               limit reached to exit loop. %                      04174
            IF topflag THEN % send out now. %                     04175
               qsender(&string, &destsw, topflag);                04176
            IF NOT dirflg THEN EXIT LOOP;                         04177
         % advance pointers for include checks and next scan %
                                                                  04178
            oldpos[1] ← endnew[1] ← nextchr;                      04179
         END; % of text/include loop %                            04180
   RETURN END.
                                                                  04181
(qmenu) PROC (stid, endstd, destsw); %output a query menu%        04182
   %Given the stid of the up of the menu and the end stid of the
   sequence, the main display area address, and a destination work
   area address, qmenu generates the output for the plex below (if
   any).                                                          04183
   Menu statements are defined as:                                04184
      All named statements                                        04185
      All statements beginning with an asterisk                   04186
   NOTE:  All other statements will appear exactly where they
   occur, interspersed with menu selections, and at same level.
                                                                  04187
   These are formatted so they become selectable by number.  Qmenu
   attaches a number to each and pushes an address to that place
   (an stid) onto the query state stack so that the mapping from
   option selection to address in a file is easy.                 04188
   Qspecs determine whether COLUMNATION is to be performed.       04189
   System globals:  qmenumax - max # simultaneous menu items
   permitted.  qcolwidth = # horizontal increments to a
   menucolumn. %                                                  04190
   LOCAL                                                          04191
      menusize,    % size of the menu before checking with user %
                                                                  04192
      menumax,     % local temp for max allowed items %           04193
      code,        % result of testing for menu statement %       04194
      stringadr,   % temp for calls %                            04195
      menusw,      % sequence work area for MENU PLEX only %      04196
      oldview[2],     % temp for saving/restoring viewspecs %     04197
      origlevel;   % level of the UP of this plex (for comparison
      to detect end)%
```

```
                                                                    04198
LOCAL STRING                                                        04199
   columnatestr [75],    %one line%                                 04200
   selection[20];         %selectable number accompanying menu
   item%                                                            04201
REF menusw, destsw;
                                                                    04202
% initialization %                                                  04203
   menumax ← menusize ← IF qspecs.queropts THEN qspecs.queropts
   ELSE qmenumax;                                                   04204
   *columnatestr* ← NULL;                                           04205
% open a new sequence for this menu plex - get first stid %
                                                                    04206
   &menusw ← openseq (stid, endstd, destsw.swvspec,
   destsw.swvsp2, 0 %no user sequence generator, please! %,
   destsw.swcacode); %open a secondary sequence so we can
   control reading of menu plex right here %                        04207
   ON SIGNAL ELSE closeseq(&menusw);                                04208
   oldview ← destsw.swvspec;                                        04209
   oldview[1] ← destsw.swvsp2;                                      04210
   stid ← seqgen (&menusw);                                         04211
      %throw away top statement - already processed %               04212
   destsw.swvspec.vstrnc ← 1;                                       04213
      % 1 line (truncated for simple menu case %                    04214
   origlevel ← menusw.swvspec.vslev ← menusw.swclvl + 1;            04215
      % 1 level (truncated for simple menu case % % change to
      permit menued lower levels if desired. %                      04216
% process the entire plex %                                        04217
   %-----set viewspecs = 1 line%                                    04218
   LOOP %once for each statement%                                   04219
      BEGIN                                                         04220
      stid ← seqgen(&menusw); %next stid%                           04221
      %test for done (or no data)%                                  04222
         IF stid = endfil THEN EXIT;                                04223
         CASE menusw.swclvl OF                                      04224
            < origlevel: EXIT LOOP; % done with menu plex %
                                                                    04225
            > origlevel: REPEAT LOOP; % sub of plex level %
                                                                    04226
            ENDCASE NULL: % fall through %                          04227
      %see if it's a menu statement%                                04228
         CASE code ← tstmenu(stid) OF                               04229
            = 1, = 2: % menu statements %                           04230
               BEGIN                                                04231
               %test for too many menu items - exceeds system
               max%                                                 04232
                  IF (qmenucnt ← qmenucnt + 1) > menumax THEN
                                                                    04233
                     BEGIN                                          04234
                     % Go back to interact with user; must come
                     back here to close menusw or continue. %
                                                                    04235
                     destsw.swstid ← endfil;                        04236
                     sport(&destsw);                                04237
                     menumax ← mencont(menumax, menusize,
                     &menusw, &destsw, $oldview);                   04238
```

```
        END:                                                    04239
    % Put the item on the menu stack. %                         04240
      IF newstk THEN                                            04241
        BEGIN                                                   04242
        IF NOT pushent(stid, mentyp) THEN                       04243
          BEGIN                                                 04244
          % Cannot SIGNAL or use err because of
          switched stacks.  Rather, set global
          error flag to be TRUE, and send back
          endfil. %                                             04245
          hseqer ← TRUE;                                        04246
          % send back EOF because we are done %
                                                                04247
            closeseq(&menusw);                                  04248
            destsw.swcstid ← destsw.swstid ←
            endfil;                                             04249
            sport(&destsw); %don't want to be
            called again.  Finished with a query
            branch%                                             04250
          END;                                                  04251
        END;                                                    04252
    % build selection number string %                           04253
      *selection* ← STRING(qmenucnt), '., SP;                   04254
    IF qspecs.quercol THEN %columnation requested%
                                                                04255
      BEGIN                                                     04256
      IF code = 2 THEN                                          04257
        BEGIN                                                   04258
        % This is an unnamed, menued statement in
        the middle of a columnated menu. %                      04259
        % Flush out any current columnation stuff
        and continue with the first line of this
        menued, unnamed statement. %                            04260
        IF columnatestr.L THEN                                  04261
          qcolumnate(&destsw, stid,
          $columnatestr, $selection, TRUE);                     04262
        % Continue processing. %                               04263
        END                                                     04264
      ELSE                                                      04265
        BEGIN                                                   04266
        qcolumnate(&destsw, stid, $columnatestr,
        $selection, FALSE); %add this%                          04267
        EXIT CASE;                                              04268
        END;                                                    04269
      END;                                                      04270
    %straight menu%                                             04271
      qstmt(&menusw, &destsw, FALSE, $selection);
                                                                04272
        %do the statement, executing INCLUDES, no
        substructure, append "selection" to front%
                                                                04273
    IF overscreen THEN                                          04274
      BEGIN                                                     04275
      menumax ← mencont(menumax, menusize, &menusw,
      &destsw, $oldview);                                       04276
      % reset overscreen %                                      04277
```

```
                        overscreen ← FALSE;                          04278
                     END;                                            04279
                  END; % of menu-handling %                          04280
               = 3, = 5: % non-menu statements - may have
               directives %                                          04281
                  BEGIN                                              04282
                  %flush out any current columnation stuff%          04283
                  %process it, exclude substructure%                 04284
                     stringadr ← IF code = 3 THEN $"" ELSE %
                     asterisk case % $selection;                     04285
                     % truncation may be on - make default number
                     of lines show for this case %                   04286
                        destsw.swvspec.vstrnc ← oldview.vstrnc;
                                                                     04287
                     qstmt(&menusw, &destsw, FALSE, stringadr);
                                                                     04288
                     IF overscreen THEN                              04289
                        BEGIN                                        04290
                        menumax ← mencont(menumax, menusize,
                        &menusw, &destsw, $oldview);                 04291
                        overscreen ← FALSE;                          04292
                        END;                                         04293
                     % restore the temporary suppression of line
                     truncation disabled for non-named menu or
                     non-menu %                                      04294
                        destsw.swvspec.vstrnc ← 1;                   04295
                  END;                                               04296
               = 0: NULL; % watch out - error occurred %             04297
               ENDCASE NULL;                                         04298
            END; %of menu plex loop%
                                                                     04299
      %cleanup%                                                      04300
         %flush out possible incomplete line%                        04301
            IF columnatestr.L THEN                                   04302
               qcolumnate(&destsw, stid, $columnatestr, $selection,
               TRUE);                                                04303
            closeseq(&menusw); %done with menu plex sequence.  Rgturn
            to caller%                                               04304
         destsw.swvspec ← oldview;                                   04305
         destsw.swvsp2 ← oldview[1];                                 04306
      RETURN; %normal exit%                                          04307
      END.
                                                                     04308
   (mencont) PROC (menumax, menusize, menusw, destsw, oldview);      04309
      REF menusw, destsw, oldview;                                   04310
      IF moremen THEN                                                04311
         BEGIN                                                       04312
         menumax ← menumax + menusize; % on with the next chunk %
                                                                     04313
         END                                                         04314
      ELSE                                                           04315
         BEGIN                                                       04316
         closeseq(&menusw);                                          04317
         destsw.swvspec ← oldview;                                   04318
         destsw.swvsp2 ← oldview[1];                                 04319
         destsw.swstid ← destsw.swcstid ← endfil;                    04320
```

```
      sport(&destsw);                                                  04321
      END;                                                             04322
   RETURN(menumax);                                                    04323
   END.
                                                                       04324
(tstmenu) PROC (stid); %test if query menu stmt%                       04325
   % RETURNs                                                          04326
      0: if ERROR encountered                                          04327
      1: if statement represented by stid is named (therefore
      menu)                                                            04328
      2:  if statement is an unnamed menued statement                  04329
      3:  if statement represented by stid starts with non-menu
      symbol                                                           04330
      4:  if statement begins with comment character                   04331
      5:  No "non-menu" character defined.  This is non-menu, but
      no character peeled off. %                                       04332
   LOCAL char;                                                         04333
   ON SIGNAL ELSE RETURN(0);                                           04334
   % WATCH OUT - GETNMF CALLS ERR with "Bad statement identifier"%
                                                                       04335
   IF getnmf(stid) THEN RETURN(1);                                     04336
   %search for an asterisk, pretend string is one char long%           04337
      CCPOS SF(stid);                                                  04338
      IF (char ← READC) = comchr THEN RETURN(4);                       04339
      IF NOT menchr THEN RETURN(5);                                    04340
      CASE char OF                                                     04341
         =menchr: RETURN(3);                                           04342
         ENDCASE RETURN(2);                                            04343
   END.
                                                                       04344
(qcolumnate) PROC (sw, stid, columnatestr, selection,
flushflg);%query columnation%                                          04345
   % Given the address of a sequence work area, the "current" line
   string, the selection string for the next item, and a flag for
   just flushing out the previous information, qcolumnate either
   does the simple flush, or adds the new item (sw.stid) to the
   columnated menu.
   Obeys global qcolwidth.%
                                                                       04346
      %NOTE:  sequence work area is destination -- PRIMARY one%
                                                                       04347
   REF sw, selection, columnatestr;                                    04348
   LOCAL STRING names[30];
                                                                       04349
   LOCAL savlvl, qcolwork[7]; %temp READC work area%
                                                                       04350
   %see if time to pump out current line%                              04351
      IF columnatestr.L >= 72 OR flushflg THEN                         04352
         BEGIN                                                         04353
         savlvl ← sw.swclvl := sw.swslvl + 1;                          04354
         send(&sw, $columnatestr);                                     04355
         sw.swclvl ← savlvl;                                           04356
         *columnatestr* ← NULL;                                        04357
         IF flushflg THEN RETURN;                                      04358
         END;                                                          04359
   %construct new entry%                                               04360
```

```
      %set up for and do name extraction%                                04361
         *names* ← NULL;                                                 04362
         qcolwork ← stid;                                                04363
         qcolwork[1] ← 1; %point to first char%                          04364
         fechcl(forward, $qcolwork);                                     04365
         xtrnam($names, $qcolwork, -1, 0);                               04366
      %add on to current line being built%                               04367
         *columnatestr* ← *columnatestr*, *selection*, *names*,
         *spacestr*[1 TO (qcolwidth - names.L - selection.L)];
                                                                         04368
   RETURN END.
                                                                         04369
(qlnkspec) PROCEDURE (tptadr, usrstg, fnmstg, stnstg, vspstg);
                                                                         04370
   % Makes use of the new lnkprs%                                       04371
   %--------------------%                                              04372
   LOCAL datastr[30];                                                    04373
   REF  usrstg, fnmstg, stnstg, vspstg, tptadr;                          04374
   lnkprs(&tptadr, $datastr);                                            04375
   % Set up text pointers from datastr. %                               04376
      p1 ← datastr[us]; p1[1] ← datastr[us + 1];                         04377
      p2 ← datastr[ue]; p2[1] ← datastr[ue + 1];                         04378
      p3 ← datastr[fs]; p3[1] ← datastr[fs + 1];                         04379
      p4 ← datastr[fe]; p4[1] ← datastr[fe + 1];                         04380
      p5 ← datastr[ds]; p5[1] ← datastr[ds + 1];                         04381
      p6 ← datastr[de]; p6[1] ← datastr[de + 1];                         04382
      p7 ← datastr[vb]; p7[1] ← datastr[vb + 1];                         04383
      p8 ← datastr[ve]; p8[1] ← datastr[ve + 1];                         04384
      tptadr ← datastr[le]; tptadr[1] ← datastr[le + 1];                 04385
   *usrstg* ← p1 p2; % user %                                            04386
      astruc(&usrstg);                                                   04387
   *fnmstg* ← p3 p4; % file name %                                       04388
      astruc(&fnmstg);                                                   04389
   *stnstg* ← p5 p6; % statement name, number, or marker %               04390
      IF stnstg.L > empty AND *stnstg*[1] NOT= '# THEN
      astruc(&stnstg);                                                   04391
   *vspstg* ← p7 p8; % view %                                            04392
   IF usrstg.L # empty AND fnmstg.L # empty THEN                         04393
      *fnmstg* ← '<, *usrstg*, '>, *fnmstg*                              04394
   ELSE IF NOT tptadr.stastr AND fnmstg.L # empty THEN                   04395
      BEGIN                                                              04396
      <NLS, IOEXEC, gdftdir>( tptadr.stfile, &usrstg);                   04397
      *fnmstg* ← '<, *usrstg*, '>, *fnmstg*;                             04398
      END;                                                               04399
   IF stnstg.L = empty AND fnmstg.L # empty  THEN *stnstg* ← '0;
                                                                         04400
      %May really want it to be empty, e.g., to change viewspecs
      when not changing file%                                            04401
   RETURN;                                                               04402
   END.
                                                                         04403
                                                                         04404
(ofstid)  %evaluate  intra-file address expression%                      04405
   %Given the display area address in order to get the file number
   and a string containing an address expression, this routine
   will open the file, and return the corresponding stid and
```

```
character count.%                                                    04406
%--------------------%                                              04407
PROCEDURE(dpa, stnstg);                                              04408
LOCAL vs1, vs2, cacode, useqgen;                                     04409
LOCAL TEXT POINTER stptr, z1, z2;                                    04410
REF dpa, stnstg;                                                     04411
stptr ← origin;                                                      04412
stptr.stfile ← dpa.dacsp.stfile;                                     04413
FIND SF(*stnstg*) ↑z1 SE(*stnstg*) ↑z2;                              04414
vs1 ← caddexp($z1, $z2, &dpa, $stptr : vs2, cacode, useqgen);
                                                                     04415
RETURN(stptr, stptr[1], vs1, vs2, cacode, useqgen);                  04416
END.
                                                                     04417
%  %                                                                 04418
```

```
   (feedsw)  PROCEDURE(sw, astrng);                                    04419
      %Given the address of a sequence work area, this routine
      changes its vspecs in accord with the specifications in the
      A-string passed it.  It passes the characters in the A-string
      to <PRMSPC, SETLT>, except for content analyzer patterns.%
                                                                       04420
      %-------------%                                                  04421
      LOCAL count, length, char, vs1, vs2;                             04422
      LOCAL TEXT POINTER tp;                                           04423
      REF sw, astrng;                                                  04424
      length ← astrng.L;                                               04425
      count ← empty - 1;                                               04426
      vs1 ← sw.swvspec;                                                04427
      vs2 ← sw.swvsp2;                                                 04428
      UNTIL (count ← count+1) > length DO                              04429
         IF (char ← *astrng*[count]) = '; THEN                         04430
            BEGIN                                                      04431
            UNTIL (char ← *astrng*[count ← count + 1]) = '; DO         04432
               BEGIN                                                   04433
               % skip over ca pattern. %                               04434
               IF count >= length THEN EXIT LOOP1;                     04435
               END;                                                    04436
            END                                                        04437
         ELSE vs1 ← setlt(char, vs1, vs2 : vs2);                       04438
      sw.swvspec ← vs1;                                                04439
      sw.swvsp2 ← vs2;                                                 04440
      RETURN;                                                          04441
      END.
                                                                       04442
                                                                       04443
% Context stack building %                                            04444
   (pushent) PROCEDURE (stid, type);                                   04445
      % Makes use of the global curstk as the context to be filled in
      %                                                                04446
      % Type = 1: mentyp; =2: multyp. %                                04447
      LOCAL stkad, itemad, wrkcnt, work;                               04448
      REF stkad, itemad;                                               04449
      IF NOT curstk.stktyp THEN                                        04450
         % get a new stack %                                           04451
         BEGIN                                                         04452
         % get first block in this stack; link it to conrng %          04453
         IF NOT (&stkad ← getblk(67B, &qstorblk)) THEN                 04454
            RETURN(FALSE);                                             04455
         stkad[1] ← &qstorblk;                                         04456
         curstk.constk ← &stkad ← &stkad + 2;                          04457
         stkad.conlnk ← -1;                                            04458
         stkad.stkcnt ← 0;                                             04459
         curstk.stktyp ← type;                                         04460
         END                                                           04461
      ELSE                                                             04462
         BEGIN                                                         04463
         % Use totcnt to get to the proper block. %                    04464
         &stkad ← curstk.constk;                                       04465
         FOR wrkcnt ← curstk.totcnt DOWN stkmax UNTIL <= stkmax DO
                                                                       04466
            BEGIN                                                      04467
```

```
            IF (&stkad  ← stkad.conlnk)<=0 THEN RETURN(FALSE);   04468
            END;                                                 04469
         END;                                                    04470
      % Put the entry on the stack. %                            04471
         IF (stkad.stkcnt←stkad.stkcnt+1) > stkmax THEN          04472
            BEGIN                                                04473
            %must get a new block%                               04474
            % Reset stkad.stkcnt %                              04475
               BUMP DOWN stkad.stkcnt;                           04476
            % Try to get a new block.  Put address in previous
            block's link field. %                                04477
            IF NOT (work ← getblk(67B, &qstorblk)) THEN          04478
               RETURN(FALSE);                                    04479
            stkad.conlnk ← work + 2;                             04480
            &stkad ← work;                                       04481
            stkad[1] ← &qstorblk;                                04482
            &stkad ← &stkad + 2;                                 04483
            stkad.conlnk ← -1;                                   04484
            stkad.stkcnt ← 1;                                    04485
            END;                                                 04486
         &itemad ← &stkad + stkad.stkcnt*2;                      04487
         itemad ← stid;                                          04488
         itemad[1] ← IF getnmf(stid) THEN getnam(stid)           04489
            ELSE 0; % zero if no name %                          04490
         BUMP curstk.totcnt;                                     04491
      RETURN(TRUE);                                              04492
      END.
                                                                 04493
% Storage allocation %                                          04494
FINISH;                                                           0894
```