```
< NINE, SEQGEN.NLS;6, >, 15-Aug-78 11:02 HGL ;;;;
FILE seqgen  % <ARCSUBSYS>XL10 to <RELNINE>SEQGEN %% (arcsubsys,xL10,)
(RELNINE,seqgen.rel,) %

ALLOW!
% Routines making up the sequence generator %

   % OPENSEQ is used to open a sequence
     CLOSESEQ is used to close a sequence
     SEQGEN is used to get the next statement in a sequence
     There is also a routine SEQNXT which gets the next statement in a
   sequence considering all the viewspecs except the content analyzer
   ones %

   (sqsksz) CONSTANT = 1765B;  % seqgen stack size %                    3C
   REF tda; % global pointing to display area for tty %
   (stackdecsize) CONSTANT = 10;                                        3E
% the Sequence Generator %

   (openseq) PROCEDURE    % open a sequence %                          4B
      (fstid,    % STID with which to begin the sequence %
      lbstid,    % STID of statement heading last branch in
                   the sequence%
      vspec,     % first word of viewspecs to use %
      vspc2,     % 2nd word of viewspecs %
      usqcod,    % address of user seqgen routine -- or 0 %
      cacode);   % address of CONAN routine  -- or 0 %

      % OPENSEQ
         Gets a sequence work area (with its attendent stack and
         statement vector work area) and returns the address of that
         work area.
         Initializes the work area
         If generating statement numbers, initializes the statement
         vector work area also.
         Perhaps calls the user'S seqgen (with the address of the work
         area and telling it that its being called as openseq)
         The stack is not switched which means that user openseq is not
         allowed to do port-sends. %

      LOCAL sw;
      REF sw;

      % allocate a work area, a stack, and a statement vector work
      area: call alostk if necessary%
         &sw _ getsgw();
      % initialize the work area: swbport has already been set %
         IF fstid.stastr  THEN   % stid points to a string %
            BEGIN
            sw.swstid  _ sw.swcstid _ fstid;
            sw.swclvl  _ 0;
            sw.swvspec _ vspec;
            sw.swvsp2  _ vspc2;
            END
         ELSE
            BEGIN
```

```
      IF getsid (fstid) = 0 THEN   % illegal stid %
         BEGIN
         sw.swcstid.stfile _ fstid.stfile;
         sw.swcstid.stpsid _ origin;
         sw.swstid _ endfil;
         END
      ELSE sw.swstid _ sw.swcstid _ fstid;
      % set up level and statement number stuff %
      sw.swclvl _
         IF vspec.vsstnf THEN stvect (sw.swcstid, sw.swsvw)
         ELSE getlev (sw.swcstid);
      % set up viewspecs -- maybe relative level stuff %
         sw.swvspec  _
            IF vspec.vsrlev THEN reslev (vspec, sw.swclvl)
            ELSE vspec;
         sw.swvsp2  _ vspc2;
      END;
   sw.swlbstid _ lbstid;
   sw.swusqcod _ usqcod;
   sw.swcacode _ cacode;
   sw.swslvl   _ sw.swclvl;   % remember what level started at %
   sw.swkflg   _ sw.swword _ FALSE;   % nothing has been sent or
   passed yet %
% maybe call the users seqgen code as openseq %
   IF vspec.vsusqf AND sw.swusqcod THEN
      [sw.swusqcod] (&sw, sqopn);
RETURN (&sw);
END.

(closeseq) PROCEDURE   % close a sequence %                          4C
   (sw);   % address of the sequence work area to close %           4C1

   % CLOSESEQ
      Check for a legal sequence work area.
      Perhaps calls the user'S seqgen (with the address of the work
      area and telling it that its being called as a closeseq)
      Releases the sequence work area (with its attendent stack and
      statement vector work area)
      The stack is not switched which means the user closeseq is not
      allowed to do port-sends %
   REF sw;
   % maybe call the users seqgen code as closeseq %
      IF sw.swvspec.vsusqf AND sw.swusqcod THEN
         [sw.swusqcod] (&sw, sqcls);
   % release the work area, stack, and statement vector work area:
   call dalostk %
      relsgw (&sw);
   RETURN;
   END.

(seqgen) PROCEDURE                                                   4D
   (sw REF);   % address of a sequence generator work area %

   % SEQGEN
      Calling SEQGEN results in the caller being port-sent (looks
      like a normal return to the caller) the next "STID" in that
```

```
        sequence -- actually it may be an ENDFIL (no more items in
        this sequence), or a stastr, or the STID of a statement.
        Also the following fields of the work area are updated:
           SWCSTID (remains unchanged if an ENDFIL or stastr is the
           item returned)
           SWSTID (will be same as thing returned)
           SWSRSAV and SWMRSAV; SWCLVL; SWCALL, SWKFLG
           (unless conan code is using sport rather than using send or
           returning with flag set).
        It checks for a legal sequence work area.
        It also updates the statement vector if necessary.
        SEQGEN is actually merely a dispatch routine -- it calls
        either usqcod (user sequence code) or SSEQGEN (system sequence
        generator);
        It effects one-half of the coroutine linkage involved in
        port-sends (SPORT effects the other half).
        The input fork is informed that the Sequence Generator will
        handle rubouts.  Since the only way out of here is thru SPORT,
        the occurance of a rubout is checked for there and an ENDFIL
        sent if one occured.
        The stacks are switched here.  Thus no locals are allowed and
        the argument must be saved in a global. %

    (stid);                                                          4D3
    (portb);                                                         4D4
    (erresult); % TRUE if error propogated from sw's stack %         4D5
    (erarray ) REF; % address of 4 word signal values and parmeters %
                                                                     4D6
    IF sw.swstid = endfil THEN RETURN (endfil);
    rubabt _ FALSE;  % tells input fork not to abort if see rubout %
    portb _ sw.swbport;
    stid _ PCALL [portb](&sw:[portb] erresult, &erarray);
    sw.swbport _ portb;
    IF erresult THEN
       ABORT( erarray, erarray[1], erarray[2], erarray[3] );
    RETURN(stid);
    END.

(sysqco) COROUTINE;                                                   4E
    (sw) REF;                                                        4E1
    (porta);                                                         4E2
    (s2);                                                            4E3
    (s3);                                                            4E4
    (s4);                                                            4E5
    (erarray) [4];                                                   4E6
    PORT ENTRY
       BEGIN
       INVOKE (sqgerr,sproper);
       END
    EXIT
       &sw _ PCALL %[porta]% (:[porta]);
    sw.swaport _ porta;
    LOOP
       BEGIN
       IF sw.swusqcod AND sw.swvspec.vsusqf THEN
          [sw.swusqcod](&sw, sqgnxt)
```

```
      ELSE sseqgen (&sw);
      % ought never to return here!  Sports are made in the user
      sequence generator or in sseggen.  It does the necessary
      PCALL. %
         sport(&sw);
      END;
   (sproper):                                                    4E11
      % Propogate error to other stack: PCALL not allowed in
      catchphrase %
         &sw _ PCALL [porta] ( (sw.swstid _ endfil), TRUE %error%,
         $erarray : [porta]);
         sw.swaport _ porta;
         DROP(ALL);
         ABORT( programbug, $"Should not return after final signal
         propogation in SEQGEN");
   (sqgerr) CATCHPHRASE(:s2,s3,s4);                              4E12
      CASE SIGNALTYPE OF
         =aborttype:
            BEGIN
            % propogate error to owning stack %
               porta _ sw.swaport;
               erarray _ SIGNAL;
               erarray[1] _ s2;
               erarray[2] _ s3;
               erarray[3] _ s4;
               TERMINATE; % will do a PCALL back %
            % should not get back here.  if we do, CONTINUE-- system
            base catchphrase will cause syscrv to be called. %
               CONTINUE;
            END;
         =helptype: RESUME(nohelp);
         ENDCASE CONTINUE;
   END.

(sseqgen) PROCEDURE    % NOBODY BUT SEQGEN SHOULD CALL THIS ROUTINE
%                                                                4F
   (sw);   % address of a sequence generator work area %         4F1

   % SSEQGEN
      Given address of sequence work area this procedure returns the
      next item in that sequence.
      SSEQGEN or SEQNXT or SEND or ucacod stores that value in
      SWSTID (and SWCSTID if it is not ENDFIL or a stastr).
      The routine SEQNXT finds the next STID, considering all
      viewspecs except content anlysis ones -- SSEQGEN takes care of
      those.
      Note that SEQNXT is not called the first time thru this
      routine.  Thus the first stid returned will be the one the
      sequence was initialized with -- unless conan code fails it.
      Rubouts are checked for after every call on conan code. %

   LOCAL stid;
   REF sw;
   LOOP
      BEGIN
      WHILE (sw.swspec.vsconf OR (sw.swspec.vscakf AND NOT
```

```
      sw.swkflg)) AND sw.swcacode > 0 AND sw.swstid # endfil DO
         % (conan must pass statements viewspec is on OR (turn off
         conan after first passed statement viewspec is on BUT
         haven't passed one yet)) AND there is a conan program AND
         we haven't reached the end of the sequence yet %
         BEGIN
         stid _ sw.swcstid;
         FIND SF(stid);
         IF [sw.swcacode] (&sw) OR inptrf %rubout% THEN
            BEGIN
            sw.swstid _ stid;  % a send from this same statement
            will have blown this field %
            sport (&sw);
            END;
         seqnxt (&sw);
         END;
      sport (&sw);
      seqnxt (&sw);
      END;
   END.

(seqnxt) PROCEDURE                                                  4G
   (sw);    % address of sequence work area %                      4G1

   % SEQNXT
      Returns the next STID in this sequence (or ENDFIL if no more).
      It updates the following fields of the work area:
         swstid, swcstid (if not ENDFIL), and swclvl.
      It also updates the statement vector if necessary.
      It takes into account all viewspecs, except content analysis,
      during the search. %

   LOCAL stid, %current stid%
      vspec;   %first viewspec word (sw.swvspec)%
   REF sw;
   IF sw.swcstid.stastr THEN RETURN (sw.swstid _ endfil);
   vspec _ sw.swvspec;
   IF sw.swclvl >= vspec.vslev
      OR (stid _ getsub (sw.swcstid)) = sw.swcstid THEN
      BEGIN %see if superstructure fits viewspecs%
      IF (stid _ sw.swcstid) = sw.swlbstid THEN
         RETURN (sw.swstid _ endfil);
      WHILE getftl (stid) DO
         BEGIN
         stid _ getsuc (stid);
         BUMP DOWN sw.swclvl;
         IF stid = sw.swlbstid THEN
            RETURN (sw.swstid _ endfil);
         IF vspec.vsstnf THEN stvmod (sw.swsvw, up);
         END;
      IF stid.stpsid = origin THEN RETURN (sw.swstid _ endfil);
      %successor is next PSID in sequence%
      stid _ getsuc (stid);
      IF vspec.vsstnf THEN stvmod (sw.swsvw, suc);
      END
   ELSE
```

```
      BEGIN   %substructure fits viewspecs%
      BUMP sw.swclvl;   %increase level%
      IF vspec.vsstnf THEN stvmod (sw.swsvw, sub);
      END;
   RETURN (sw.swcstid _ sw.swstid _ stid);
   END.

(seqend) PROCEDURE (stid, vs1, vs2);                                    4H
   % This procedure may be used to find the last branch to be
   handled by the sequence generator depending on the branch or plex
   only viewspecs;  the sequence generator itself no longer checks
   the veiwspecs.  Thus, the stid of the last branch must be passed
   as a parameter to openseq! %
   LOCAL retstd;
   IF stid.stastr THEN RETURN(endfil);
   retstd _ CASE TRUE OF
      = vs1.vsbrof: stid;
      = vs1.vsplxf : getail(stid);
      ENDCASE endfil;
   RETURN(retstd);
   END.

(send) PROCEDURE                                                        4I
   (sw,    % address of a sequence work area %
   str);   % an ENDFIL or address of a string %

   % SEND
      Takes the place of the old send.
      Sets the swstid (but not the swcstid) field of the work area.
      Calls sport. %

   REF sw, str;
   IF &str = endfil THEN sw.swstid _ endfil
   ELSE
      BEGIN
      sw.swstid.stastr _ TRUE;
      sw.swstid.stadr  _ &str;
      IF str.L = empty THEN sw.swstid.stadr _ $" "; %no null strings
      permitted%
      END;
   sport (&sw);
   RETURN;
   END.

(sport)  PROCEDURE    % send-port mechanism %                           4J
   (sw REF);    % address of a sequence work area %

   % SPORT
      Effects one-half of the coroutine linkage involved in
      port-sends (SEQGEN effects the other half).
      Control o'S and rubouts are checked for here.
      The stacks are switched here. %

   (porta);                                                             4J3
   sw.swkflg _ TRUE; % something has been returned in this sequence
   %
```

```
   IF inptrf THEN % a control o for or a rubout has occured %
      sw.swstid _ endfil;
   porta _ sw.swaport;
   &sw _ PCALL [porta] (sw.swstid, FALSE %No error%:[porta]);
   sw.swaport _ porta;
   RETURN;
   END.

(sqinit) PROCEDURE;   % sequence generator init %                    4K

   % SQINIT
      Is only called at INIT time.
      It initializes all the sequence work areas (including the
      attached stacks and sequence vector work areas). %

   LOCAL sw, swcnt;    % address of a sequence generator work area %
   REF sw;
   swcnt _ 0;
   FOR &sw _ $sqgwas UP $sqwrkl UNTIL >= $sqgaend DO
      BEGIN
      sw.swalloc _ FALSE;   % allocation bit %
      IF swcnt < sqstkn THEN
         BEGIN
         sw.swstkdec _ TRUE;
         sw.swstkalloc _ FALSE;
         sw.swsvw _ $sqsvws + (swcnt * (svmxlev + 1));
         IF NOT (sw.swstkloc _ getblk(sqsksz+stackdecsize, $dspblk)
         ) THEN
            ABORT( programbug, $"No room in zone for sequence
            stack");
         sw.swstksize _ sqsksz;
         END
      ELSE
         BEGIN
         sw.swstkdec _ FALSE;
         sw.swstkalloc _ FALSE;
         sw.swsvw _ sw.swstkloc _ sw.swstksize _ 0;
         END;
      BUMP swcnt;
      END;
   RETURN;
   END.

(getsgw) PROCEDURE;   % get a sequence generator work area %        4L

   % GETSGW
      Allocates, initializes some parts of, and returns the address
      of a sequence generator work area (a sequence work area, a
      statement vector work area, and a stack). %

   LOCAL sw, swcnt, portb;
   REF sw;
   swcnt _ 0;
   FOR &sw _ $sqgwas UP $sqwrkl UNTIL >= $sqgaend DO
      IF NOT sw.swalloc THEN
         BEGIN
```

```
        sw.swalloc _ TRUE;   % allocation bit %
        IF NOT sw.swstkdec THEN
           BEGIN
           IF NOT (sw.swstkloc _ getblk(sgsksz+stackdecsize,
           $dspblk) ) THEN
              ABORT( programbug, $"No room in zone for sequence
              stack");
           sw.swstkalloc _ TRUE;
           sw.swstksize _ sgsksz;
           IF NOT (sw.swsvw _ getblk(svmxlev+1, $dspblk)) THEN
              ABORT (programbug, $"no more sequence work areas");
           END;
        %OPENPORT on other stack%
           alostk( sw.swstkloc, sw.swstksize, 0 % not allocated by
           runtime%, $sysqco, 0 : [portb]);
        sw.swbport _ portb;
        RETURN (&sw);
        END
     ELSE BUMP swcnt;
  ABORT (programbug, $"no more sequence work areas");
  END.

(relsgw) PROCEDURE    % release a sequence generator work area %   4M
  (sw REF);     % address of a sequence generator work area %

  % RELSGW
     Deallocates a sequence generator work area (a sequence work
     area, a statement vector work area, and a stack). %

  IF NOT (&sw IN [$sggwas, $sggaend]
     AND ((&sw - $sggwas) MOD $sgwrkl) = 0 AND sw.swalloc) THEN
        ABORT (programbug, $"illegal sequence work area");
  dalostk( sw.swstkloc+stackdecsize);
  IF sw.swstkalloc THEN
     BEGIN
     IF NOT freeblk((sw.swstkloc),$dspblk) THEN
        ABORT (programbug, $"unable to free sequence work area
        stack");
     IF NOT freeblk((sw.swsvw),$dspblk) THEN
        ABORT (programbug, $"unable to free sequence work area
        statement vector");
     sw.swstkalloc _ FALSE;
     sw.swstkloc _ sw.swsvw _ 0;
     END;
  sw.swalloc _ FALSE;
  RETURN;
  END.

% SEQGEN, STATEMENT NUMBER / VECTOR / SIGNATURE UTILITY ROUTINES %
  (cpysw) PROCEDURE  % copy parts of one sequence work area to
  another%                                                    5A
     (fromsw, tosw);    % record pointers to work areas %
        % this has the effect of making the tosw point to the same
        item as the fromsw %
     REF fromsw, tosw;
```

```
    tosw.swcstid _ fromsw.swcstid;
    tosw.swstid  _ fromsw.swstid;
    tosw.swclvl  _ fromsw.swclvl;
    IF tosw.swvspec.vsstnf THEN mvbfbf (fromsw.swsvw, tosw.swsvw,
    svmxlev);
    RETURN;
    END.

(pjseqg) PROCEDURE (sw, which); % Print Journal Seq Generator %    5B
    LOCAL pjvs1, pjvs2, pjcacode, pjuseqg, pjopnusqc, sv1, sv2,
    adstr[40], pjmstid;
    LOCAL TEXT POINTER pjtp;
    LOCAL STRING pjstn [200];
    REF sw, pjsw, pjlsw;
    CASE which OF
       =sqopn: % called at open %
          BEGIN
          pjrbab _ rubabt; % save current RUBOUT disposition %
          rubabt _ FALSE; % and disable RUBOUT %
          pjrubout _ FALSE; % init RUBOUT indicator %
          pjopnusqc _ IF pjsavf THEN pjusqc ELSE 0; % assure user seq
          gen control (if any ) if appropriate viewspec on %
          &pjsw _ openseq (sw.swcstid, sw.swlbstid, sw.swvspec,
          sw.swvsp2, pjopnusqc, sw.swcacode); % open secondary
          sequence %
          pjstid _ FALSE; % record not in linked-to file %
          RETURN; % return to caller %
          END;
       =sqgnxt: NULL; % called for next in seq -- fall through %
       =sqcls: % called at close %
          BEGIN
          closeseq (&pjsw); % close secondary sequence %
          IF pjstid THEN % if in a linked-to file %
             BEGIN
             closeseq (&pjlsw); % close that sequence %
             close (pjstid.stfile); % and it'S file %
             END;
          rubabt _ pjrbab; % restore RUBOUT disposition %
          RETURN; % and return to caller %
          END;
       ENDCASE err (); % called for any other purpose -- error %
    (rptloop):                                                    5B6
    LOOP % here for each statement in primary file %
       BEGIN
       IF pjrubout THEN sw.swstid _ endfil % if RUBOUT hit, force
       endfil %
       ELSE
          BEGIN
          seqgen (&pjsw); % fetch next in primary file %
          cpysw (&pjsw, &sw); % copy for caller %
          END;
       sport (&sw); % return it to him %
       pjtp _ pjsw.swstid; % build text pointer %
       pjtp[1] _ 1; % to start of statement %
       IF FIND SF(pjtp) ["Location: "] ^pjtp *( [*)] THEN
```

```
IF (pjmstid_getsub(pjtp)) # pjtp THEN
   BEGIN %if message already there, don't go after link%
   pjmstid _ getail(pjmstid);
   IF FIND SF(pjmstid) ("Message:") THEN REPEAT LOOP;
   END;
INVOKE (sigloop, rptloop);
lnkpspc (1, $pjtp, $stno, $stn2, $pjstn, $num, $adstr); %
parse the link %
pjvs1 _ tda.davspec; % save the display %
pjvs2 _ tda.davspc2; % area viewspecs %
pjcacode _ tda.dacacode;  %and other fields%
pjuseqg _ tda.dausqcod;   %that get stored into%
pjstid _ nfstid ($stn2, $pjstn, $num, &tda); % open the
secondary file %
tda.davspec _ pjvs1; % restore the display %
tda.davspc2 _ pjvs2; % area viewspecs %
tda.dacacode _ pjcacode;
tda.dausqcod _ pjuseqg;
DROP (sigloop);
tda.davspec.vsbrof _ FALSE;
feedlt (&tda, $num); % build viewspecs from link %
pjopnusqc _ IF pjsavf THEN pjusqc ELSE 0; % assure user seq
gen control (if any ) if appropriate viewspec on %
&pjlsw _ openseq (pjstid, seqend(pjstid, sv1 _ tda.davspec,
sv2 _ tda.davspc2), sv1, sv2, pjopnusqc, sw.swcacode); %
open tertiary sequence using them %
tda.davspec _ pjvs1; % restore the display %
tda.davspc2 _ pjvs2; % area viewspecs again %
IF NOT pjrubout THEN
   BEGIN
   pjsep (&sw, $"Text of Cited Document Follows"); % output
   document header %
   LOOP % here for each cited document %
      BEGIN
      IF pjrubout THEN % if RUBOUT hit %
         BEGIN
         pjrubout _ FALSE; % reset RUBOUT indicator %
         send (&sw, $"..."); % show that the document was
         aborted %
         EXIT LOOP; % and be finished with it %
         END;
      seqgen (&pjlsw); % fetch next of cited document %
      IF pjlsw.swstid = endfil THEN % if end of document %
         BEGIN
         rubabt _ FALSE; % we know that 'sport' reset
         'rubabt' %
         EXIT LOOP; % end of tertiary sequence %
         END;
      cpysw (&pjlsw, &sw); % if no RUBOUT, return next %
      sport (&sw); % of tertiary sequence to caller %
      END;
   pjsep (&sw, $"End of Cited Document"); % output document
   trailer %
   END;
closeseq (&pjlsw); % close tertiary sequence %
close (pjstid,gtfile); % and its file %
```

```
      pjstid _ FALSE; % mark no tertiary seq in progress %
      END;
   END;

(sigloop) CATCHPHRASE;                                          5B9
   CASE SIGNAL OF
      =errsig: % if any error is encountered %
         BEGIN
         DISABLE (sigloop);
         pjsep (&sw, $"Document has been Archived"); % Assume the
         file is not on-line %
         TERMINATE; % and abandon it %
         END;
      ENDCASE
         CONTINUE;

END.

(pjsep) PROCEDURE (sw, title); % Print Journal Header/Trailer %    5C
   REF sw, title;
   LOCAL len, i, j, end;
   LOCAL STRING str [300];
   % get horizontal increment and line length from display area %
      len _ tda.damcol-hinc; % length of print line %
   % force the sequence generator to make this line a level 1 thing
   %
      sw.swclvl _ 1; % force statement level to highest %
   % edit the separator string %
      % start with a blank line %
         *str* _ EOL;
      % left of title %
         end _ (j_len-(2+title.L)*hinc)/2;
         FOR i _ hinc UP hinc UNTIL >= end DO
            *str* _ *str*, '<;
      % title %
         *str* _ *str*, SP, *title*, SP;
      % right of title %
         end _ len -2*hinc;
         FOR i _ i+title.L*hinc + 2*hinc UP hinc UNTIL >= end DO
            *str* _ *str*, '>;
      % lower left corner %
         *str* _ *str*, EOL;
   % send the string to the seq. gen. %
      send (&sw, $str); % return header/trailer to caller %
   RETURN;
   END.

(reslev) PROCEDURE(vspec, clevel);                              5D
   %Given a viewspec word and a level, this routine will perform any
   relative level adjustment required by the viewspec word and
   return the updated viewspec word.%
   IF vspec.vsrlev THEN
      BEGIN
      IF vspec.vslevd THEN % up %
         vspec.vslev _ MAX (clevel-vspec.vslev, 1)
```

```
        vspec.vslev _ MIN (clevel+vspec.vslev, 63);
      vspec.vsrlev _ FALSE;
      vspec.vslevd _ FALSE;
      END;
   RETURN(vspec);
   END.
```

(stvect) 5E

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

```
   %-----------------%
   PROCEDURE (stid, stvwrk);
   LOCAL
      curwrd, %current word being built in vector%
      level;  %level of STID%
   REF stvwrk, curwrd;
   &curwrd _ &stvwrk + svmxlev;
   level _ 0;
   UNTIL stid.stpsid = origin DO %count up to origin%
      BEGIN
      IF (&curwrd _ &curwrd-1) <= &stvwrk THEN err(5);
      curwrd _ stpos(stid);
      stid _ getup(stid);
      BUMP level;
      END;
   stvwrk _ level;
   %move vector to top of work area%
   mvbfbf (&curwrd, &stvwrk+1, level);
   RETURN (level);
   END.
```

(stvmod) 5F

%This routine modifies a statement vector. It is called with the address of a statement vector as the first argument, and a type parameter as the second. The statement vector is assumed to be initialized, and is modified as specified by TYPE.%

```
   %-----------------%
   PROCEDURE (stvwrk, type);
   LOCAL word; %current word in statement vector%
   REF stvwrk, word;
   IF stvwrk = 0 THEN   % treat the origin statement specially %
      CASE type OF
         =sub:
            BEGIN
```

```
            stvwrk _ 1;
            [&stvwrk + 1] _ 1;
            END;
         =suc, =pred, =up: NULL;
         ENDCASE err(0)
   ELSE
      BEGIN
      &word _ &stvwrk + stvwrk;
      CASE type OF
         =sub:
            BEGIN
            IF (stvwrk _ stvwrk + 1) > svmxlev THEN err(5);
            [&word + 1] _ 1;
            END;
         =suc:  word _ word + 1;
         =pred: word _ MAX (word-1, 1);
         =up:   stvwrk _ MAX (stvwrk-1, 0);
         ENDCASE err(0);
      END;
   RETURN;
   END.

(stpos)                                                          5G
   %Given a STID, this routine returns an integer which is the
   position of that statement within its plex.%
   %-----------------%
   PROCEDURE (stid);
   LOCAL
      nstid,
      posit; %current position%
   IF getfhd(stid) THEN RETURN (1);
   posit _ 1;
   nstid _ gethed(stid);
   UNTIL nstid = stid DO
      BEGIN
      nstid _ getsuc(nstid);
      BUMP posit;
      END;
   RETURN (posit);
   END.


(getlev)                                                         5H
   %Called with STID, returns level of that statement.%
   %-----------------%
   PROC(stid);
   LOCAL level; %current level%
   level _ 0;
   UNTIL stid.stpsid = origin OR getorf( stid ) DO
      BEGIN
      stid _ getup(stid);
      BUMP level;
      END;
   RETURN (level);
   END.
```

```
(fechno)                                                              5I
   %Appends statement number of stid to string.  Give the STID as
   the first argument, and the address of the string which is to
   contain the statement number as the second.  The statement
   number will be built in the string.  If the structure is not
   intact or the statement vector cannot be built, a call to RERROR
   or an EXCEED CAPICITY ERROR may result.%
   %-----------------%
   PROCEDURE(stid, astr);
   LOCAL fchsvw[100];
   stvect(stid, $fchsvw);
   fechnm($fchsvw, astr);
   RETURN;
   END.

(fechnp)                                                              5J
   %This is like FECHNO, but generates the statement number of the
   successor (whether it exists or not).%
   %-----------------%
   PROCEDURE(stid, astr);
   LOCAL fchsvw[50];
   stvect(stid, $fchsvw);
   stvmod($fchsvw, suc);
   fechnm($fchsvw, astr);
   RETURN;
   END.

(fechnd)                                                              5K
   %This is like FECHNO, but generates the statement number of the
   sub (whether it exists or not).%
   %-----------------%
   PROCEDURE(stid, astr);
   LOCAL fchsvw[50];
   stvect(stid, $fchsvw);
   stvmod($fchsvw, sub);
   fechnm($fchsvw, astr);
   RETURN;
   END.

(fechnm) PROCEDURE (stvwrk, astr);                                    5L
   %This routine will append the statement number to the string,
   given the statement vector.  The address of the first word of the
   statement vector is provided as the first argument, and the
   address of the A-string for the statement number is given as the
   second.
   The algorithm used is roughly as follows:
      The field of the statement vector corresponding to the
      current level is divided by a base.
         Two bases are used: 10 (for digits) and 26 (for
         letters and &).
      There is no alphabetic zero (complicating the algorithms.)
      The quotient is added to an appropriate offset to create
      an ascii character, and this character is then added
      to the A-string.
```

```
      "powers" have been exhausted.  In the case of alphabetic
      fields, we must then go back through the local string
      assembled and translate "zero" characters into strings
      containing only the letters a through z.
      The base is then changed and the statement vector field
      for the next level is then divided as above.%
   %-----------------%
   LOCAL
      base, %current base%
      basnxt, % next base %
      curlev, %current level%
      posit, %current plex position%
      power, %power of base in plex postion = number of
         characters in statement number for this plex%
      char, %next character in statement number%
      len, % flag used when converting alpha fields %
      zflag, % flag used when converting alpha flags: zero character
      has been encountered. %
      offset, %converts number indicating position
         to number/alpha character%
      offnxt,  %next offset%
      alpharray[10];
   REF astr, stvwrk;
   IF stvwrk = 0 THEN
      BEGIN
      *astr* _ *astr*, "0;
      RETURN;
      END;
   (base, basnxt, offset, offnxt) _                                    5L7
      (numbase, alphbase, numoff, alphoff);
   curlev _ 0;
   LOOP
      BEGIN
      IF (curlev := curlev + 1) >= stvwrk THEN RETURN;
      posit _ stvwrk[curlev];
      %find largest power of base contained in plex position%
      power _ 1;
      IF base=numbase THEN
         BEGIN
         WHILE power*base <= posit DO power _ power*base;
         %now put out the actual string%
         UNTIL power < 1 DO
            BEGIN
            DIV posit / power, char, posit;
            *astr* _ *astr*, char + offset;
            power _ power / base;
            END;
         END
      ELSE % alphabase %
         BEGIN
         WHILE power*base < posit DO power _ power*base;
         alpharray _ 0;
         UNTIL power < 1 DO
            BEGIN
            DIV posit / power, char, posit;
```

```
            alpharray[alpharray] _ char;
            power _ power / base;
            END;
         % alpharray now has 0s rather than 26s (Zs); we must go
         through the array to find them and convert them (and their
         neighbors if necessary).  Finally we have an array which
         may be used to produce characters to be appended to the
         passed astr. %
            zflag _ FALSE;
            FOR len _ alpharray DOWN UNTIL <= 0 DO
               BEGIN
               IF zflag := FALSE THEN
                  alpharray[len] _ alpharray[len] - 1;
               IF alpharray[len] <= 0 THEN
                  BEGIN
                  IF len=1 THEN EXIT LOOP;
                  alpharray[len] _ alpharray[len] + 26;
                  zflag _ TRUE;
                  END;
               END;
            len _ IF alpharray[1] > 0 THEN 1 ELSE 2;
            FOR len UP UNTIL > alpharray DO
               *astr* _ *astr*, alpharray[len]+offset;
         END;
      (base, basnxt, offset, offnxt) _                              5L9H
         (basnxt, base, offnxt, offset);                            5L9H1
      END;
   END.
(fechux) PROCEDURE                                                  5M
   %Given the address of an a-string as the first argument and a
   file number as the second, this routine returns the STID of the
   statement whose statement number is contained in the A-string.
   If the statement does not exist (or FECHUX is passed an
   illegal statement number) it returns a -1.
   The algorthm used is roughly as follows:
      The plex position indicated by each (alphabetic or
      numeric) level field in the statement number is converted
      to an integer, then the psid of the statement
      corresponding to that position is found.
      LODCHR is used to read characters from the statement
      number, and the resulting ascii character is converted
      to a number.
      This number is calculated by subtracting an appropriate
      offset (20B for numbers and 40B for lettrs), then
      multiplying the difference by an appropriate base (10
      for digits and 26 for letters).
      This number is added to any plex position already
      calculated.
      This subtraction, multiplication, and addition continues
      until a new level in the statement number is
      encountered.%
   %-----------------%
   (astr, fileno);
   LOCAL stid, %current stid%
      curlen, %position current character in statemet number%
```

```
      base, %base for converting character to number%
      basnxt, %next base%
      zero, %zero point for converting character to number%
      zeronxt,  %next zero%
      posit; %current plex position%
   REF astr;
   posit _ 0;
   stid _ origin;
   stid.stfile _ fileno;
   curlen _ 1;
   base _ numbase;
   basnxt _ alphbase;
   zero _ numoff;
   zeronxt _ alphoff;
   LOOP
      BEGIN
      LOOP %read characters, calculate position%
         BEGIN
         curc _ *astr*[curlen];
         IF curc IN ['a, 'z] THEN curc _ curc - upcase;
         % check if have gone from alpha to digit or vice versa
         %
         IF (base=numbase AND (curc _ curc-zero) NOT IN [0, base))
         OR (base=alphbase AND (curc _ curc-zero) NOT IN (0, base])
         THEN EXIT;
         posit _ posit * base + curc;
         IF (curlen _ curlen+1) > astr.L THEN EXIT;
         END;
      %get stid in position indicated by posit, one level down
      from stid%
      IF posit = 0 THEN RETURN
         (IF stid.stpsid = origin THEN stid ELSE endfil);
      IF (stid := getsub(stid)) = stid THEN RETURN (endfil);
      UNTIL (posit _ posit-1) = 0 DO
         BEGIN
         IF getftl(stid) THEN RETURN (endfil);
         stid _ getsuc(stid);
         END;
      IF curlen > astr.L THEN RETURN (stid);
      (base, basnxt, zero, zeronxt, posit) _                    5M17I
         (basnxt, base, zeronxt, zero, 0);                      5M17I1
      END;
   END.


(techsig) %***%                                                    5N
   PROCEDURE (stid, astrng); %append statement signature to string%
   %assumes real statement--not an a-string%
   %----------------%
   LOCAL sdbadr, word, bytptr, count, char, stdb;
   REF sdbadr, astrng;
   % eventually want to change this to work for any property type %
      IF NOT lodprop( stid, txttyp :&sdbadr, stdb) THEN
         err($" No text block with this node");
   % initials %
```

```
        bytptr _ stbptr(empty) + $word;
        count _ 0;
        UNTIL (count _ count + 1) > 4 DO
           IF (char _ ^bytptr) = 0 THEN EXIT LOOP
           ELSE *astrng* _ *astrng*, char;
        *astrng* _ *astrng*, SP;
     % date and time %
        dtfrmt(sdbadr.stime, &astrng);
     RETURN;
     END.
```

FINISH of seggen

(closeseq)	<nine, seqgen, 075>	PROCEDURE	4C
(cpysw)	<nine, seqgen, 0307>	PROCEDURE	5A
(technd)	<nine, seqgen, 0579>	LOCAL	5K
(technm)	<nine, seqgen, 0929>	PROCEDURE	5L
(techno)	<nine, seqgen, 0558>	LOCAL	5I
(techno)	<nine, seqgen, 0568>	LOCAL	5J
(techsig)	<nine, seqgen, 0713>	LOCAL	5N
(technx)	<nine, seqgen, 01018>	PROCEDURE	5M
(getlev)	<nine, seqgen, 0543>	LOCAL	5H
(getsgw)	<nine, seqgen, 0769>	PROCEDURE	4L
(openseq)	<nine, seqgen, 015>	PROCEDURE	4B
(pjsep)	<nine, seqgen, 0426>	PROCEDURE	5C
(pjseqg)	<nine, seqgen, 0319>	PROCEDURE	5B
(relsgw)	<nine, seqgen, 0804>	PROCEDURE	4N
(reslev)	<nine, seqgen, 0455>	PROCEDURE	5D
(send)	<nine, seqgen, 0211>	PROCEDURE	4I
(seqend)	<nine, seqgen, 0201>	PROCEDURE	4H
(seqgen)	<nine, seqgen, 092>	PROCEDURE	4D
(seqnxt)	<nine, seqgen, 0160>	PROCEDURE	4G
(sport)	<nine, seqgen, 0232>	PROCEDURE	4J
(sqinit)	<nine, seqgen, 0738>	PROCEDURE	4K
(sqsksz)	<nine, seqgen, 09>	CONSTANT =1765B	3C
(sseqgen)	<nine, seqgen, 0129>	PROCEDURE	4F
(stackdecsize)	<nine, seqgen, 0892>	CONSTANT =10	3E
(stpos)	<nine, seqgen, 0524>	LOCAL	5G
(stvect)	<nine, seqgen, 0468>	LOCAL	5E
(stvmod)	<nine, seqgen, 0492>	LOCAL	5F
(sysqco)	<nine, seqgen, 0861>	COROUTINE	4E

(tezone)	<nine, spdata, 024>	EXT	3D
(tsblend)	<nine, spdata, 05>	EXT	3C
(tsblist)	<nine, spdata, 04>	EXT	3B

```
< NINE, SPDATA.NLS;6, >, 5-APR-77 18:15 SKO ;;;;
FILE spdata % <ARCSUBSYS>XL10  to <RELNINE>spdata%% <arcsubsys,xl10,>
TO <relnine,spdata.rel,>%
%This is data that must be on page boundary known to the FE and BE in
the shared page protocol version of NLS9.  The structure of this file
cannot be changed without changing all reference to it in both the FE
and BE.%
% free storage block for lists %
   %DO NOT CHANGE ORDER OF THE FOLLOWING TWO DECLARES%
   (fsblist) EXTERNAL [17777B];                                        3B
   (fsblend) EXTERNAL;                                                 3C
   (rezone) EXTERNAL [10000B];                                         3D
%global message lists.  BE sndlst is FE rcvlist.  BE rcvlst is FE
sndlst.  format of list message is a LIST with 4 elements:
   LIST(                                                               4A
      INDEX, type with value call or acknowledgement                   4A1
      STRING/INDEX,  procedure name/error code - 0 error code means
      successful                                                       4A2
      INDEX,  tid - 0 means no acknowledgement required                4A3
      LIST  argument/return                                            4A4
      )%
         (sndlst) EXTERNAL LIST [4];  %message list for BE call on FE%
         (rcvlst) EXTERNAL LIST [4];  %message list for FE call on BE%
      % FILL UP THE REST OF THE PAGE %
         (spfiller) EXTERNAL [770B];
            %CHANGE THIS ARRAY SIZE IF OTHER VARIABLES ADDED%

FINISH                                                                 6
```

(cabts)	<nine, srecords, 082>	LOCAL	3B
(ccchar)	<nine, srecords, 0300>	FIELD - 7	3A3
(ccdefine)	<nine, srecords, 0297>	RECORD	3A
(ccdevice)	<nine, srecords, 0298>	FIELD - 7	3A1
(ccecho)	<nine, srecords, 0301>	FIELD - 7	3A4
(cctype)	<nine, srecords, 0299>	FIELD - 7	3A2
(chars)	<nine, srecords, 086>	RECORD	3C
(chnul)	<nine, srecords, 087>	FIELD - 1	3C1
(daauxiliary)	<nine, srecords, 0343>	FIELD - 1	3D40
(dabottom)	<nine, srecords, 0338>	FIELD - 18	3D36
(dacacode)	<nine, srecords, 0351>	FIELD - 18	3D48
(daccnt)	<nine, srecords, 0314>	FIELD - 12	3D12
(daccol)	<nine, srecords, 0324>	FIELD - 18	3D22
(dacrow)	<nine, srecords, 0321>	FIELD - 18	3D19
(dacsp)	<nine, srecords, 0312>	FIELD - 36	3D10
(dacurallo)	<nine, srecords, 0355>	FIELD - 18	3D52
(daempty)	<nine, srecords, 0332>	FIELD - 1	3D30
(daexis)	<nine, srecords, 0331>	FIELD - 1	3D29
(dafrzi)	<nine, srecords, 0315>	FIELD - 18	3D13
(daind)	<nine, srecords, 0317>	FIELD - 18	3D15
(daleft)	<nine, srecords, 0333>	FIELD - 18	3D31
(dalftjst)	<nine, srecords, 0320>	FIELD - 18	3D18
(dalink)	<nine, srecords, 0327>	FIELD - 18	3D25
(damcol)	<nine, srecords, 0326>	FIELD - 18	3D24
(damind)	<nine, srecords, 0318>	FIELD - 18	3D16
(damrow)	<nine, srecords, 0323>	FIELD - 18	3D21
(dapstf)	<nine, srecords, 0330>	FIELD - 5	3D28
(dapvs)	<nine, srecords, 0308>	FIELD - 36	3D6
(dapvs2)	<nine, srecords, 0310>	FIELD - 36	3D8
(daright)	<nine, srecords, 0335>	FIELD - 18	3D33
(daseq)	<nine, srecords, 0344>	FIELD - 1	3D41
(dastoid)	<nine, srecords, 0342>	FIELD - 18	3D39
(dastrt)	<nine, srecords, 0340>	FIELD - 18	3D37
(dasuppress)	<nine, srecords, 0345>	FIELD - 1	3D42
(datab0)	<nine, srecords, 0347>	FIELD - 36	3D44
(datab1)	<nine, srecords, 0348>	FIELD - 36	3D45
(datab2)	<nine, srecords, 0349>	FIELD - 36	3D46
(datop)	<nine, srecords, 0336>	FIELD - 18	3D34
(daukeycod)	<nine, srecords, 0354>	FIELD - 18	3D51
(dausqcod)	<nine, srecords, 0352>	FIELD - 18	3D49
(davspc2)	<nine, srecords, 0306>	FIELD - 36	3D4
(davspec)	<nine, srecords, 0304>	FIELD - 36	3D2
(dawid)	<nine, srecords, 0329>	FIELD - 9	3D27
(dblock)	<nine, srecords, 0401>	RECORD	3G
(dbnxt)	<nine, srecords, 0403>	FIELD - 18	3G2
(dbtype)	<nine, srecords, 0407>	FIELD - 18	3G6
(delst)	<nine, srecords, 0405>	FIELD - 18	3G4
(denxt)	<nine, srecords, 0404>	FIELD - 18	3G3
(displayarea)	<nine, srecords, 0302>	RECORD	3D
(dlgacc)	<nine, srecords, 035>	FIELD - 1	2B2
(dlgars)	<nine, srecords, 036>	FIELD - 1	2B3
(dlgart)	<nine, srecords, 038>	FIELD - 1	2B5
(dlgbyt)	<nine, srecords, 039>	FIELD - 1	2B6
(dlgdar)	<nine, srecords, 037>	FIELD - 1	2B4
(dlgdcr)	<nine, srecords, 040>	FIELD - 1	2B7
(dlgddm)	<nine, srecords, 042>	FIELD - 1	2B9

(dlgdfr)	<nine, srecords, 044>	FIELD - 1	2B11
(dlgdlt)	<nine, srecords, 041>	FIELD - 1	2B8
(dlgdmt)	<nine, srecords, 043>	FIELD - 1	2B10
(dlgdov)	<nine, srecords, 046>	FIELD - 1	2B13
(dlgdrd)	<nine, srecords, 048>	FIELD - 1	2B15
(dlgdwr)	<nine, srecords, 049>	FIELD - 1	2B16
(dlgiwr)	<nine, srecords, 045>	FIELD - 1	2B12
(dlgprt)	<nine, srecords, 047>	FIELD - 1	2B14
(dlgrp)	<nine, srecords, 033>	RECORD	2B
(dlgrvr)	<nine, srecords, 034>	FIELD - 1	2B1
(dliacc)	<nine, srecords, 08>	FIELD - 1	2A3
(dliars)	<nine, srecords, 09>	FIELD - 1	2A4
(dliart)	<nine, srecords, 010>	FIELD - 1	2A5
(dlibyt)	<nine, srecords, 014>	FIELD - 1	2A9
(dlidfr)	<nine, srecords, 012>	FIELD - 1	2A7
(dlidlt)	<nine, srecords, 05>	FIELD - 2	2A1
(dlidmt)	<nine, srecords, 011>	FIELD - 1	2A6
(dlifrf)	<nine, srecords, 07>	FIELD - 1	2A2
(dliiwr)	<nine, srecords, 013>	FIELD - 1	2A8
(dlimis)	<nine, srecords, 015>	FIELD - 1	2A10
(dlinex)	<nine, srecords, 020>	FIELD - 1	2A15
(dlinto)	<nine, srecords, 04>	RECORD	2A
(dlinrw)	<nine, srecords, 016>	FIELD - 1	2A11
(dlinvr)	<nine, srecords, 019>	FIELD - 1	2A14
(dliprt)	<nine, srecords, 017>	FIELD - 1	2A12
(dlisiz)	<nine, srecords, 018>	FIELD - 1	2A13
(dlitar)	<nine, srecords, 021>	FIELD - 2	2A16
(dlitcr)	<nine, srecords, 023>	FIELD - 2	2A17
(dlitdm)	<nine, srecords, 025>	FIELD - 2	2A18
(dlitov)	<nine, srecords, 027>	FIELD - 2	2A19
(dlitrd)	<nine, srecords, 029>	FIELD - 2	2A20
(dlitwr)	<nine, srecords, 031>	FIELD - 2	2A21
(dlsacc)	<nine, srecords, 052>	FIELD - 1	2C2
(dlsalp)	<nine, srecords, 053>	FIELD - 1	2C3
(dlsart)	<nine, srecords, 054>	FIELD - 1	2C4
(dlsbyt)	<nine, srecords, 056>	FIELD - 1	2C6
(dlsdfr)	<nine, srecords, 061>	FIELD - 1	2C11
(dlsdlt)	<nine, srecords, 058>	FIELD - 1	2C8
(dlsdmt)	<nine, srecords, 060>	FIELD - 1	2C10
(dlslen)	<nine, srecords, 063>	FIELD - 1	2C13
(dlsiwr)	<nine, srecords, 062>	FIELD - 1	2C12
(dlsnac)	<nine, srecords, 064>	FIELD - 1	2C14
(dlsnrd)	<nine, srecords, 065>	FIELD - 1	2C15
(dlsnwr)	<nine, srecords, 066>	FIELD - 1	2C16
(dlsort)	<nine, srecords, 050>	RECORD	2C
(dlsrvr)	<nine, srecords, 051>	FIELD - 1	2C1
(dlssiz)	<nine, srecords, 069>	FIELD - 1	2C19
(dlstar)	<nine, srecords, 055>	FIELD - 1	2C5
(dlstcr)	<nine, srecords, 057>	FIELD - 1	2C7
(dlstdm)	<nine, srecords, 059>	FIELD - 1	2C9
(dlstov)	<nine, srecords, 067>	FIELD - 1	2C17
(dlstrd)	<nine, srecords, 068>	FIELD - 1	2C18
(dlstwr)	<nine, srecords, 070>	FIELD - 1	2C20
(dparad)	<nine, srecords, 0406>	FIELD - 18	3G5
(dprev)	<nine, srecords, 0402>	FIELD - 18	3G1
(dpsrds)	<nine, srecords, 0250>	FIELD - 36	3P1

(ercondir)	<nine, srecords, 0253>	FIELD - 36	3P4
(erdata)	<nine, srecords, 0268>	FIELD - 36	3P19
(erdatime)	<nine, srecords, 0257>	FIELD - 36	3P8
(erdlen)	<nine, srecords, 0267>	FIELD - 36	3P18
(erfns1)	<nine, srecords, 0261>	FIELD - 36	3P12
(erfns2)	<nine, srecords, 0262>	FIELD - 36	3P13
(erfns3)	<nine, srecords, 0263>	FIELD - 36	3P14
(erfns4)	<nine, srecords, 0264>	FIELD - 36	3P15
(erfns5)	<nine, srecords, 0265>	FIELD - 36	3P16
(erfns6)	<nine, srecords, 0266>	FIELD - 36	3P17
(erhost)	<nine, srecords, 0251>	FIELD - 36	3P2
(erjobn)	<nine, srecords, 0255>	FIELD - 18	3P6
(erline)	<nine, srecords, 0254>	FIELD - 18	3P5
(erlpbaudf)	<nine, srecords, 0258>	FIELD - 18	3P9
(erlptype)	<nine, srecords, 0259>	FIELD - 18	3P10
(erprom)	<nine, srecords, 0260>	FIELD - 36	3P11
(errordoc)	<nine, srecords, 0249>	RECORD	3P
(ertipn)	<nine, srecords, 0256>	FIELD - 36	3P7
(eruser)	<nine, srecords, 0252>	FIELD - 36	3P3
(frexis)	<nine, srecords, 0165>	FIELD - 1	3H2
(frhexis)	<nine, srecords, 0168>	FIELD - 1	3I3
(frhlast)	<nine, srecords, 0170>	FIELD - 6	3I2
(frhtop)	<nine, srecords, 0169>	FIELD - 6	3I1
(frrentry)	<nine, srecords, 0163>	RECORD	3H
(frrheader)	<nine, srecords, 0167>	RECORD	3I
(frsring)	<nine, srecords, 0164>	FIELD - 18	3H1
(gbhst)	<nine, srecords, 0172>	RECORD	3J
(nstnmf)	<nine, srecords, 0175>	FIELD - 10	3J3
(nstnmi)	<nine, srecords, 0173>	FIELD - 18	3J1
(nstnmn)	<nine, srecords, 0174>	FIELD - 8	3J2
(iptmchr)	<nine, srecords, 0242>	RECORD	3O
(oc2qtr)	<nine, srecords, 0271>	FIELD - 9	3Q2
(ochar1)	<nine, srecords, 0275>	FIELD - 1	3Q6
(ochar2)	<nine, srecords, 0274>	FIELD - 3	3Q5
(ochar3)	<nine, srecords, 0273>	FIELD - 3	3Q4
(ocjnk1)	<nine, srecords, 0270>	FIELD - 18	3Q1
(ocjnk2)	<nine, srecords, 0272>	FIELD - 2	3Q3
(octchar)	<nine, srecords, 0269>	RECORD	3Q
(srcc)	<nine, srecords, 0203>	FIELD - 12	3K2
(srexis)	<nine, srecords, 0204>	FIELD - 1	3K3
(srhexis)	<nine, srecords, 0212>	FIELD - 1	3L3
(srhfileno)	<nine, srecords, 0210>	FIELD - 5	3L2
(srhfname)	<nine, srecords, 0209>	FIELD - 18	3L1
(srhlast)	<nine, srecords, 0214>	FIELD - 6	3L5
(srhtop)	<nine, srecords, 0213>	FIELD - 6	3L4
(srpsid)	<nine, srecords, 0201>	FIELD - 18	3K1
(srrheader)	<nine, srecords, 0208>	RECORD	3L
(srvs1)	<nine, srecords, 0205>	FIELD - 36	3K4
(srvs2)	<nine, srecords, 0206>	FIELD - 36	3K5
(statt)	<nine, srecords, 0380>	FIELD - 4	3F6
(stbcoord)	<nine, srecords, 0385>	FIELD - subfields	3F11
(stbpe)	<nine, srecords, 0399>	FIELD - 36	3F32
(stbps)	<nine, srecords, 0397>	FIELD - 36	3F30
(stcbug)	<nine, srecords, 0394>	FIELD - 3	3F24
(stccnt)	<nine, srecords, 0377>	FIELD - 12	3F3

(stcuralio)	<nine, srecords, 0393>	FIELD - 18	3F18
(stecoord)	<nine, srecords, 0387>	FIELD - subfields	3F13
(stexis)	<nine, srecords, 0379>	FIELD - 1	3F5
(stflag)	<nine, srecords, 0378>	FIELD - 1	3F4
(stkeep)	<nine, srecords, 0417>	FIELD - 1	3F28
(stlev)	<nine, srecords, 0383>	FIELD - 6	3F9
(stlsid)	<nine, srecords, 0391>	FIELD - 9	3F23
(stlsptr)	<nine, srecords, 0410>	FIELD - 18	3F16
(stnew)	<nine, srecords, 0395>	FIELD - 1	3F25
(stoid)	<nine, srecords, 0413>	FIELD - 18	3F22
(strepo)	<nine, srecords, 0414>	FIELD - 1	3F26
(strngid)	<nine, srecords, 0390>	FIELD - 9	3F20
(strngtab)	<nine, srecords, 0374>	RECORD	3F
(stselector)	<nine, srecords, 0381>	FIELD - 8	3F7
(stsrce)	<nine, srecords, 0382>	FIELD - 4	3F8
(ststid)	<nine, srecords, 0375>	FIELD - 36	3F1
(stwid)	<nine, srecords, 0389>	FIELD - 9	3F19
(stx2)	<nine, srecords, 0386>	FIELD - 18	3F15
(tmchr1)	<nine, srecords, 0248>	FIELD - 6	3O6
(tmchr2)	<nine, srecords, 0247>	FIELD - 6	3O5
(tmchr3)	<nine, srecords, 0246>	FIELD - 6	3O4
(tmchr4)	<nine, srecords, 0245>	FIELD - 6	3O3
(tmchr5)	<nine, srecords, 0244>	FIELD - 6	3O2
(tmchr6)	<nine, srecords, 0243>	FIELD - 6	3O1
(view2specs)	<nine, srecords, 0240>	RECORD	3N
(viewspecs)	<nine, srecords, 0217>	RECORD	3N
(vsaflf)	<nine, srecords, 0232>	FIELD - 1	3M15
(vsblkf)	<nine, srecords, 0227>	FIELD - 1	3M10
(vsbrof)	<nine, srecords, 0225>	FIELD - 1	3M8
(vscakf)	<nine, srecords, 0235>	FIELD - 1	3M18
(vscapf)	<nine, srecords, 0223>	FIELD - 1	3M6
(vsdatf)	<nine, srecords, 0238>	FIELD - 1	3M21
(vsfrzf)	<nine, srecords, 0234>	FIELD - 1	3M17
(vsidtf)	<nine, srecords, 0236>	FIELD - 1	3M19
(vsindf)	<nine, srecords, 0228>	FIELD - 1	3M11
(vslev)	<nine, srecords, 0219>	FIELD - 6	3M2
(vslevd)	<nine, srecords, 0222>	FIELD - 1	3M5
(vsmkrf)	<nine, srecords, 0241>	FIELD - 1	3M1
(vsnamf)	<nine, srecords, 0229>	FIELD - 1	3M12
(vspagf)	<nine, srecords, 0237>	FIELD - 1	3M20
(vsplxf)	<nine, srecords, 0226>	FIELD - 1	3M9
(vsrind)	<nine, srecords, 0233>	FIELD - 1	3M16
(vsrlev)	<nine, srecords, 0221>	FIELD - 6	3M4
(vssidf)	<nine, srecords, 0239>	FIELD - 1	3M22
(vsstnf)	<nine, srecords, 0230>	FIELD - 1	3M13
(vsstnr)	<nine, srecords, 0231>	FIELD - 1	3M14
(vstrnc)	<nine, srecords, 0220>	FIELD - 6	3M3
(vsusgf)	<nine, srecords, 0224>	FIELD - 1	3M7
(wal)	<nine, srecords, 0372>	EXT	3E15
(wamax)	<nine, srecords, 0373>	EXT	3E16
(wfrozen)	<nine, srecords, 0367>	FIELD - 1	3E10
(wiatt)	<nine, srecords, 0365>	FIELD - 4	3E8
(widdarea)	<nine, srecords, 0358>	FIELD - 18	3E1
(widexis)	<nine, srecords, 0362>	FIELD - 1	3E5
(widindex)	<nine, srecords, 0359>	FIELD - 9	3E2
(widnpar)	<nine, srecords, 0360>	FIELD - 9	3E3

(windowarea)	<nine, srecords, 0357>	RECORD	3E
(wipriority)	<nine, srecords, 0363>	FIELD - 4	3E6
(wiselector)	<nine, srecords, 0366>	FIELD - 8	3E9
(wirx)	<nine, srecords, 0371>	FIELD - 18	3E14
(wtype)	<nine, srecords, 0364>	FIELD - 3	3E7
(wupix)	<nine, srecords, 0370>	FIELD - 18	3E13

```
< NINE, SRECORDS.NLS;11, >, 27-APR-77 09:42 KJM ;;;;( MICHAEL,
SRECORDS.NLS;2, ), 16-JUL-74 09:16 EKM ;
FILE srecords % <ARCSUBSYS>XL10  <RELNINE>srecords % %
(arcsubsys,xl10,) (RELNINE,srecords.rel,) %
% records for use with the directory commands %
   (dlinfo) RECORD  % information to be listed for Directory commands %
                                                                    2A
      dlidlt[2],        % undeleted or deleted only or both %
         % 0 - undeleted only; 1 - deleted only; 2 or 3 - both %
      dlifrf[1],        % list for file group (not directory) %
      dliacc[1],        % list account of file %
      dliars[1],        % list archive status %
      dliart[1],        % list archive tapes %
      dlidmt[1],        % list dump tape %
      dlidfr[1],        % list default number of versions to keep %
      dlilwr[1],        % list last writer %
      dlibyt[1],        % list length and bytesize %
      dlimis[1],        % list miscellaneous information about file %
      dlinrw[1],        % list number of reads and writes %
      dliprt[1],        % list protection %
      dlisiz[1],        % list size in pages %
      dlinvr[1],        % don't list vesion numbers %
      dlinex[1],        % don't list extension names %
      dlitar[2],        % list [time and] date of archiving %
         % 1 - date only; 2 or 3 - time and date %
      dlitcr[2],        % list [time and] date of creation this version
      %
         % 1 - date only; 2 or 3 - time and date %
      dlitdm[2],        % list [time and] date of last dump %
         % 1 - date only; 2 or 3 - time and date %
      dlitov[2],        % list [time and] date original version created
      %
         % 1 - date only; 2 or 3 - time and date %
      dlitrd[2],        % list [time and] date of last read %
         % 1 - date only; 2 or 3 - time and date %
      dlitwr[2];        % list [time and date of last write %
         % 1 - date only; 2 or 3 - time and date %
   (dlgrp) RECORD  % grouping information for Directory commands %    2B
      dlgrvr[1],        % group in reverse order %
      dlgacc[1],        % group by accounts %
      dlgars[1],        % group by archive status %
      dlgdar[1],        % group by archive date %
      dlgart[1],        % group by archive tapes %
      dlgbyt[1],        % group by bytesize %
      dlgdcr[1],        % group by creation date %
      dlgdlt[1],        % group by deleteion status %
      dlgddm[1],        % group by dump date %
      dlgdmt[1],        % group by dump tapes %
      dlgdfr[1],        % group by file retention specs %
      dlglwr[1],        % group by last writer %
      dlgdov[1],        % group by creation date of original version %
      dlgprt[1],        % group by protection %
      dlgdrd[1],        % group by read date %
      dlgdwr[1];        % group by write date %
   (dlsort) RECORD  % sorting information for Directory commands %    2C
```

```
    dlsacc[1],          % sort by accounts %
    dlsalp[1],          % sort alphabetically %
    dlsart[1],          % sort by archive tapes %
    dlstar[1],          % sort by archive time and date %
    dlsbyt[1],          % sort by bytesize %
    dlstcr[1],          % sort by time and date of creation %
    dlsdlt[1],          % sort by deletion status %
    dlstdm[1],          % sort by time and date of dump %
    dlsdmt[1],          % sort by dump tape %
    dlsdfr[1],          % sort by file retention specs %
    dlslwr[1],          % sort by last writer %
    dlslen[1],          % sort by length in bytes %
    dlsnac[1],          % sort by number of accesses %
    dlsnrd[1],          % sort by number of reads %
    dlsnwr[1],          % sort by number of writes %
    dlstov[1],          % sort by orig. version creation time & date %
    dlstrd[1],          % sort by last read time and date %
    dlssiz[1],          % sort by size in pages %
    dlstwr[1];          % sort by last write time and date %
%Record Definitions%
  (ccdefine) RECORD % format of entries in table "cctbl" %          3A
    ccdevice[7],        % internal device code %
    cctype[7],          % cntrlchar %
    ccchar[7],          % cntrlchar psuedonym %
    ccecho[7];          % to be echoed as %
  (cabts)      %content analysis bits%                             3B
    RECORD
    cackf[1],     %true if have tested with current pattern%
    capsf[1];     %true if have passed with current pattern.%
  (chars) RECORD                                                   3C
    chnul[1], chr4[7], chr3[7], chr2[7], chr1[7], chr0[7];

  (displayarea) RECORD %entrylength = dal, max = damax%            3D
    % **** IF YOU CHANGE THIS RECORD DEFINITION, FOR SURE TELL THE
    OUTPUT PROCESSOR GUY SO HE CAN CHANGE THE COPY IN THE OUTPUT
    PROCESSOR -- BEFORE YOU BRING UP A NEW NLS!!! **** %
    davspec[36],  % SEQUENCE  first viewspec word--1st word in
    record%
    %-----%
    davspc2[36],  % SEQUENCE second viewspec word--2nd word in
    record%
    %-----%
    dapvs[36],    % SEQUENCE previous first viewspec word%
    %-----%
    dapvs2[36],   % SEQUENCE previous second viewspec word%
    %-----%
    dacsp[36],    % SEQUENCE csp for this da%
    %-----%
    daccnt[12],   % SEQUENCE character count for dacsp t-pointer%
    dafrzl[18],   % SEQUENCE a frozen statement chain for this da%
    %-----%
    daind[18],    % FORMATTER indentation per level%
    damind[18],   % FORMATTER max indentation%
    %-----%
    dalftjst[18], % ???? left-justify branch, plex for indentation
```

```
    dacrow[18],    % FORMATTER current row count%
    %----%
    damrow[18],    % FORMATTER max row count%
    daccol[18],    % FORMATTER current column count%
    %----%
    damcol[18],    % FORMATTER max column count%
    dalink[18],    % ???? a link-jump stack for this da%
    %----%
    dawid[9],   % window id for this display area %
    dapstf[5],     %file id, previous value for this da; used in a
    kludegy fashion to see if extensive reformatting is necessary
    after a jump%
    daexis[1],     %this da entry is in use%
    daempty[1],    %da-has-no-display flag%
    daleft[18],    % FRONTEND left boundry of da %
    %-----%
    daright[18],   % FRONTEND right boundry of da%
    datop[18],     % FRONTEND top boundry of da%
    %-----%
    dabottom[18], % FRONTEND bottom boundry of da%
       %(0,0) is lower left corner-- Note the variance with the
       coordinates in the corresponding window record which are
       respect to the owning window.%
    dastrt[18],    % FORMATTER string table starting address%
    %-----%
    dastold[18],    % FORMATTER old string table starting address:
    used in reformatting%
    daauxiliary[1], % FRONTEND a da used for other purposes than
    displaying.
       (as done in TNLS)  Calculator, for example, uses such an
    area.%
    daseq[1],      % FRONTEND sequential display area%
    dasuppress[1],    % FRONTEND the display image is suppressed%
    %----%
    datab0[36],    % FORMATTER first tab position word%
    datab1[36],    % FORMATTER second tab position word%
    datab2[36],    % FORMATTER third tab position word%
    %----%
    dacacode [18],    % SEQUENCE address of content analyzer program
    -- or 0 %
    dausqcod [18],    % SEQUENCE address of sequence generator program
    -- or 0 %
    %----%
    daukeycod[18],    % FRONTEND  address of sort key extractor
    program -- or 0 %
    dacurallo[18]; % "current" block address for allocation of string
    elements %
    DECLARE EXTERNAL dal = displayarea.SIZE, damax = 7;
(windowarea) RECORD % entry length is wal %                          3E
    widdarea[18], % address of corresponding displayarea or zero %
    widindex[9], % wid -- 0 if not assigned %
    widowner[9], % index of owning window--  1 if no owner %
    %-----%
    widexis[1], % window exists %
    wipriority[4], % window priority-- (suppression, auxiliary, etc?)
    %
```

```
    wtype[3], % type of window %
    wiatt[4], % default string attributes of window %
    wiselector[8], % selector type for window-- (way items may be
    selected: always, never, as text, etc.) %
    wfrozen[1], % if true, may not split the window-- frozen
    boundaries; default is false %
    %-----%
    % diagonal coordinates-- wrt owning window %
    wuplx[18], wuply[18], % upper left coordinate %
    wlrx[18],  wlry[18]; %lower right coordinate %
    (wal) EXTERNAL = windowarea.SIZE;                                  3E15
    (wamax) EXTERNAL = 15;                                             3E16
(strngtab) RECORD  %string reference table entry%                        3F
    ststid[36],    %stid of statement being displayed%
    %-------%
    stccnt[12], % ordinal relative to start of the statement of the
    first character of this line segment. %
    stflag[1], % If TRUE, this is a string segment. %
    stexis[1],     %entry exists flag %
    statt[4], % attributes of string %
    stselector[8], % selector type for string-- (way items may be
    selected: always, never, as text, etc.) %
    stsrce[4],     %source of display data (code)%
    stlev[6],      %level of this statement%
    %-------%
    stbcoord[  stx1[18], sty[18] ], % coordinates of string/line
    segment origin %
    %-------%
    stecoord[  stex[18], stey[18] ], % coordinates of string end
    (useful in calculating what must be scrolled)  This is set only
    in the first segment %
    %-------%
    stx2[18],  % x-coordinate of end of this (first) line segment %
    stisptr[18], % address of first line segment block  for this
    string or zero %
    %-------%
    stcurallo[18], % "current" block address for allocation of line
    segment elements %
    stwid[9],  % id of window in which this string lives %
    strngid[9],     %string id from FE %
    %-------%
    stold[18], %address of corresponding entry in other string table%
    stlsid[9], % ls id from FE %
    stcbug[3],     %TRUE if line segment is buggable as a single
    character; FALSE if individual characters are buggable%
    stnew[1], % TRUE if this line segment is "new" %
    strepo[1], %TRUE if line segment is repositioned on screen%
    stcomplete[1], %TRUE if string is complete on screen%
    stkeep[1],  % TRUE if the allocated string block is to be kept %
    %-------%
    stbps[36],     %starting byte pointer to text - increment and
                       load byte gets 1st byte of string%
    %----%
    stbpe[36];     %ending byte pointer to text - plus 1 char
    further%
```

```
(dblock) RECORD % block header for string or line segment 20 element
blocks; entry length is dspbhl %                                    3G
   dprev[18], %Pointer to previous (20 element) block; 0 if none %
   dbnxt[18], % link to next block %
   denxt[18], % address of next element in this block %
   delst[18], % address of last element in this block %
   dparad[18], % address of parent data structure %
   dbtype[18]; % type of block: line segment or string %
   DECLARE EXTERNAL dspbhl = dblock.SIZE;

(frrentry) RECORD  %file return ring entry%                         3H
   frsrring[18],  %address of statement return ring%
   frexis[1];     %true if exists%
   DECLARE EXTERNAL frrelen = frrentry.SIZE;  %length of file return
   ring header%
(frrheader) RECORD   %file return ring header%                      3I
   frhtop[6],  %index to top of ring%
   frhlast[6], %index to last entry%
   frhexis[1]; %true if valid frr header%
   DECLARE EXTERNAL frrhlen = frrheader.SIZE;  %length of file
   return ring header%
(gbhst) RECORD        % word returned by getab from HOSTN table %   3J
   hstnmi[18],        % index into HSTNAM table %
   hstnmn[8],         % host number for this entry %
   hstnmf[10]         % flags for this host %
   ;
(srrentry) RECORD  %statement return ring entry%                    3K
   srpsid[18],  %partial statement pointer for a statement%
      %to be combined with srhfileno to form stid%
   srcc[12],    %character count%
   srexis[1],   %true if exists%
   srvs1[36],   %first viewspec word%
   srvs2[36];   %second viewspec word%
   DECLARE EXTERNAL srrelen = srrentry.SIZE;  %length of file return
   ring entry%
(srrheader) RECORD  %header for statement return ring%              3L
   srhtname[18],  %address of file name string%
   srhfileno[5],  %nls file number for this file %
      % 0 if file not open%
   srhexis[1],    %true if valid frr header%
   srhtop[6],     %index to current top of ring%
   srhlast[6];    %index to last entry%
   DECLARE EXTERNAL
      srrhlen = srrheader.SIZE;  %length of statement return ring
      header%
(viewspecs) RECORD                                                  3M
   % **** IF YOU CHANGE THIS RECORD DEFINITION, FOR SURE TELL THE
   OUTPUT PROCESSOR GUY SO HE CAN CHANGE THE COPY IN THE OUTPUT
   PROCESSOR -- BEFORE YOU BRING UP A NEW NLS!!! **** %
   vslev[6],      %lower level bound--level clipping%
   vstrnc[6],     %line truncation value%
   vsrlev[6],     %relative level%
   vslevd[1],     %direction of relative level adjustment%
   vscapf[1],     %content analyzer pass-flag%
```

```
    vsbrof[1],     %branch only flag%
    vsplxf[1],     %plex-only flag%
    vsblkf[1],     %blank line flag%
    vsindf[1],     %indenting on flag%
    vsnamf[1],     %names on flag%
    vsstnf[1],     %loc-nums on flag%
    vsstnr[1],     %loc-nums on right flag%
    vsaflf[1],     %abbreviated feedback line flag%
    vsrind[1],     %relative indenting flag%
    vsfrzf[1],     %frozen statements flag%
    vscakf[1],     %content analyzer k viewspec flag%
    vsidtf[1],     %initials, date, and time flag%
    vspagf[1],     %page flag for tty output%
    vsdaft[1],     %display area format flag%
    vssidf[1];     %display sids flag%
(view2specs) RECORD                                                    3N
    vsmkrf[1];     %display markers flag%
(iptmchr) RECORD %time for remote display%                             3O
    tmchr6[6], %low order 6 bits%
    tmchr5[6],
    tmchr4[6],
    tmchr3[6],
    tmchr2[6],
    tmchr1[6]; %high order 6 bits%
(errordoc) RECORD  %for recording line processor errors -- used by
,inpfbk,errfill and user subsystem lpinterp%                           3P
    ercode[36], % code for type of record %
    erhost[36], % host number %
    eruser[36], % user name %
    ercondir[36], % connected directory %
    erline[18], % tty line number %
    erjobn[18], % TENEX job number %
    ertipn[36], % TIP number if any %
    erdatime[36], % time and date %
    erlpbaudf[18], % baud rate factor %
    erlptype[18], % LP type code %
    erprom[36], %LP prom number%
    erfns1[36], % file name string %
    erfns2[36],
    erfns3[36],
    erfns4[36],
    erfns5[36],
    erfns6[36],
    erdlen[36], % number of data items %
    erdata[36]; % first data word %
(octchar) RECORD %To get at the octal representations of a
character%                                                             3Q
    ocjnk1[18],
    oc2qtr[9],  %2nd quarter of a word%
    ocjnk2[2],
    ochar3[3],  %3rd char of octal representation of a char%
    ochar2[3],  %2nd char ....%
    ochar1[1];  %1st char ....%
FINISH
% We think these are not needed%                                       5A
```

```
   ckpt1[1],     % need to update first checkpoint file %
   ckpt2[1],     % need to update second checkpoint file %
   nxckpt[1];    % which checkpoint to update next %
(halfword) RECORD                                                    5B
   rhword[18], %right half word%
   lhword[18]; %left half word%

(inpsent) RECORD %format of input stack element%                     5C
   ipstid[36],  %stid, if marker, else endfil%
   ipchar[36],  %last character read, (char count, if marker)%
   ipcoords[36],  %coordinates of last character read%
   iptime[36];  %time charactr read%

   DECLARE EXTERNAL inpsel = 4;  %length of element on input stack%

(inrptrs) RECORD %pointers to input element%                         5D
   inpcur[18],  %last read by lookc%
   inpptr[18];  %last read by input%

(reghalves) RECORD %for using two halves of words%                   5E
   rh[18], %right half word%
   lh[18]; %left half word%
```

```
< L10, STGMGT.NLS.4, >, 10-Jan-78 14:56 ANDY ;;;;
FILE stgmgt   % (arcsubsys,x110,) to (l10, stgmgt,) (arcsubsys,l1011,)
to (l1011, stgmgt,)  %
(blkhdr) RECORD %  block header record%                                    2
   blklength[15],   %length of block (in words) requested%
   blkfree[1],      %true if block is on a free list%
   sysblength[15],  %actual length (in address units) of block%
   blkprevfree[1]; %true if previous block is on a free list%
%tree list pointers% %INDEXES%
   (blksuc) CONSTANT = blkhdr.SIZE;  %addr of successor block%          3A
   (blkprd) CONSTANT = blkhdr.SIZE + 1;  %addr of predecessor block%    3B
%zone header% %INDEXES%
   (wrdsused) EXTERNAL CONSTANT = 0;                                     4A
   (blkmax) EXTERNAL CONSTANT = 1;                                       4B
   (blkmin) EXTERNAL CONSTANT = 2;                                       4C
   (blkmxi) EXTERNAL CONSTANT = 3;                                       4D
   (blksused) EXTERNAL CONSTANT = 4;                                     4E
   (zonend) EXTERNAL CONSTANT = 5;                                       4F
   (zfrelst) EXTERNAL CONSTANT = 6;                                      4G

% error strings %
   %+CPU10% % for 10, use actual strings %                               5A
      (smezos) %400% = $"Storage Management Error: Zone out of space";
                                                                         5A1
      (smefib) %401% = $"Storage Management Error: Attempt to free
      invalid block";                                                    5A2
      (smebip) %402% = $"Storage Management Error: bad index passed to
      REPLENISH";                                                        5A3
      %+CPU10%                                                           5A4
   %+CPU11% % for 11, use error numbers %                                5B
      (smen) CONSTANT = 21400B;                                          5B1
      (smezos) CONSTANT = smen+0;                                        5B2
      (smefib) CONSTANT = smen+1;                                        5B3
      (smebip) CONSTANT = smen+2;                                        5B4
      %+CPU11%                                                           5B5
%storage allocator%
   (getblk) PROCEDURE  %allocate a blk of "size" words in "zone"%        6A
      (size,  %number of words desired in blk%
      zone REF);  %address of free storage zone%
      %------------------%
      %allocates a blk in zone of at least size+blkhdr.SIZE words.  The
      blk consists of a blk header and at least size words of zeros.
      The address returned is that of the begining of the blk (the blk
      header) plus blkhdr.SIZE.  There are at least size usable words
      at this address.  If no blk can be allocated, a FALSE return of
      zero is made.%
      %------------------%
      LOCAL
         blk REF,    %address of blk%
         r,       %remainder temp%
         zero,    %temp%
         end,     %temp%
         index;   %index into zones free lists for blocks of
                   correct size%
      size _ size*ADR-PER-WORD;
```

```
   IF zone[blkmin] = zone[blkmax] THEN
      IF zone[blkmax] >= size+blkhdr.SIZE * ADR-PER-WORD THEN
         index _ 0
      ELSE
         %RETURN[FALSE](0)% err(smezos)
   ELSE
      BEGIN  %round up to nearest available size%
      index _ (size + blkhdr.SIZE * ADR-PER-WORD + zone[blkmin] -
      1)/zone[blkmin] - 1;
         %index now in words%
      IF index NOT IN [0, zone[blkmxi]] THEN %RETURN[FALSE](0)%
      err(smezos);
      END;
   IF NOT zone[zfrelst + index] AND (zone[blkmin] = zone[blkmax] OR
   NOT replenish(index, &zone)) THEN
      %no free blocks of that index or larger%
      %RETURN[FALSE](0)% err(smezos);
   &blk _ zone[zfrelst + index];
   %unlink it%
      unlink(&blk, &zone);
   %zero the blk%
      zero _ &blk + blkhdr.SIZE * ADR-PER-WORD;
      end _ &blk + blk.sysblength;
      DO [zero] _ 0 UNTIL (zero _ zero + ADR-PER-WORD) >= end;
   blk.blklength _ size/ADR-PER-WORD + blkhdr.SIZE; %for caller's
   use only%
      %in WORD units%
   BUMP zone[blksused];
   zone[wrdsused] _ zone[wrdsused] + blk.sysblength;
      %actually address-units used%
   RETURN[TRUE](&blk + blkhdr.SIZE * ADR-PER-WORD);
   END.

(freeblk) PROCEDURE  %de-allocate a block in zone%                    6B
   (blk REF,  %address of block to be de-allocated%
   zone REF); %address of free storage zone%
   %----------------%
   %the block addressed by blk is marked as free.  It is not linked
   back into the appropriate free list in zone (as was done in the
   past).
   NOTE:  Freeblk will return FALSE if it finds a bad block address.
    You may run out of space if you never issue another good free to
   replace a bad call.%
   %----------------%
   LOCAL
      previous REF, %address of previous block in the zone%
                 %for combining free blocks%
      next REF;      %address of next block in zone%
                 %for combining free blocks%
   IF (&blk _ &blk - blkhdr.SIZE * ADR-PER-WORD) NOT IN [&zone,
   zone[zonend]] THEN
      RETURN[FALSE](FALSE);
   IF blk.sysblength NOT IN [zone[blkmin], zone[blkmax]] OR
   blk.blkfree THEN
      err(smefib);
```

```
   %  this is the old code to combine adjacent free blocks
   IF zone[blkmin] NOT= zone[blkmax] THEN
      BEGIN
      &next _ &blk + blk.sysblength;
      IF &next < zone[zonend] THEN
         BEGIN
         IF next.blkfree AND (blk.sysblength + next.sysblength <=
         zone[blkmax]) THEN
            BEGIN %%combine them%%
            unlink(&next, &zone);
            blk.sysblength _ blk.sysblength + next.sysblength;
            next.sysblength _ 0;
            &next _ &blk + blk.sysblength;
            END;
         END;
      IF blk.blkprevfree AND (blk.sysblength + [&blk - ADR-PER-WORD]
      <= zone[blkmax]) THEN
         BEGIN %%combine them%%
         %%last word of previous block contains the length of that
         block%%
         &previous _ &blk - [&blk - ADR-PER-WORD];
         unlink(&previous, &zone);
         previous.sysblength _ previous.sysblength + blk.sysblength;
         blk.sysblength _ 0;
         &blk _ &previous;
         END;
      END;
   %
   linkup(&blk, &zone);
   BUMP DOWN zone[blksused];
   RETURN[TRUE](TRUE);
   END.

(makezone) PROCEDURE  %make a zone out of a block%                    6C
   (zone REF,  %address of a block to be initialized as a zone%
   max,    %maximum size in words of a block -- integral multiple of
   min%
   min,    %minimum size in words of a block%
   size);  %size in words of the zone%
   %----------------%
   %This routine takes the address of a free storage block and
   constructs an appropriate header so that it will serve as a free
   storage zone.  If min and max are the same, then the zone will
   contain only fixed sized block and a single free list -- no
   combining of adjacent free blocks will be attempted by freeblk%
   %----------------%
   LOCAL
      blk REF,     %address of a block within the zone%
      lastfreelist,    %displacement to the lastfreelist free list
      for this zone%
      frstblk REF, %address of first block in the zone%
      base REF,    %address of a block within the zone%
      length;  %length of rest of block%
   % zero out this whole block before starting %
      FOR length _ (size-1) DOWN UNTIL < 0 DO
```

```
    size _ size*ADR-PER-WORD;
    min _ MAX(min, blkhdr.SIZE+3);
    max _ ((max+min-1)/min)*min;
    zone[blkmxi] _ (max/min) - 1;
    max _ max*ADR-PER-WORD; %in address units%
    min _ min*ADR-PER-WORD; %in address units%
    zone[blkmax] _ max;
    zone[blkmin] _ min;
    lastfreelist _ zone[blkmxi] + zfrelst; %index to last free list%
    &frstblk _ &base _ &zone + lastfreelist*ADR-PER-WORD +
    ADR-PER-WORD;
    length _ ((size - (&frstblk - &zone))/min)*min;
    zone[zonend] _ &frstblk + length;
    UNTIL length < max DO
       BEGIN
       &blk _ &base + max;
       base.sysblength _ max;
       [&blk-ADR-PER-WORD] _ max;
       base.blkfree _ base.blkprevfree _ TRUE;
       base[blksuc] _ zone[lastfreelist] := &base;
       base[blkprd] _ 0;
       IF &base _ base[blksuc] THEN
          base[blkprd] _ zone[lastfreelist];
       &base _ &blk;
       length _ length - max;
       END;
    frstblk.blkprevfree _ FALSE;
    IF length > 0 THEN
       BEGIN
       blk.sysblength _ length;
       IF length < min THEN
          blk.blkfree _ blk.blkprevfree _ FALSE
       ELSE
          BEGIN
          blk.blkfree _ blk.blkprevfree _ TRUE;
          blk[blksuc] _ zone[zfrelst + length/min - 1] := &blk;
          blk[blkprd] _ 0;
          IF &base _ blk[blksuc] THEN
             base[blkprd] _ &blk;
          [&blk + blk.sysblength -ADR-PER-WORD] _ length;
          END;
       END;
    zone[blksused] _ zone[wrdsused] _ 0;
    RETURN;
    END.

(replenish) PROCEDURE %replenish supply of free blocks in zone%    6D
    (index, %free list index for size blocks needed%
    zone REF);  %address of free storage zone%
    %--------------%
    %This routine will try to replenish the supply of blocks
    corresponding to freelist specified by index.  First it will try
    to find a larger block which can be subdivided to replenish the
    supply of blcks desired.  It will try to find a block which is a
    multiple in size (so all pieces will go on same free list).  If
```

```
    If no larger blocks are found, it will attempt to combine samller
    blocks into a block of the size desired.  It does this by
    searching freelists for a free block and then attempting to
    combine adjacent blocks as required.  If this is unsuccessful, it
    returns FALSE; otherwise, breakup is used to decompose the larger
    block and add to the supply of desired blocks and this routine
    returns TRUE%
    %--------------%
    LOCAL
       indexplus1, %value of index+1 (used to speed up loop)%
       blk REF,    %address of a larger free block%
       newblk REF, %addr of blk returned when combining smaller blks
       %
       size, %words needed %
       maxi,  %value of zone[blkmxi] (used to speed up loop)%
       n,     %temp, used in loop%
       i;     %temp, used as loop variable ad free list index%
    IF index NOT IN [0, zone[blkmxi]] THEN
       err(smebip);
    maxi _ zone[blkmxi];
    %try for multiples first%
       i _ indexplus1 _ index+1;
       n _ 2;
       UNTIL (i _ indexplus1*n - 1) > maxi DO
          BEGIN
          IF zone[i + zfrelst] THEN %got one%
             BEGIN
             unlink(&blk _ zone[i+zfrelst], &zone);
             breakup(&blk, &zone, index);
             RETURN(TRUE);
             END;
          BUMP n;
          END;
    %Find any%
       i _ index;
       UNTIL (i _ i + 1) > maxi DO
          IF zone[zfrelst + i] THEN
             BEGIN %got one%
             &blk _ zone[zfrelst+i];
             unlink(&blk, &zone);
             breakup(&blk, &zone, index);
             RETURN(TRUE);
             END;
    %Try to combine smaller adjacent blocks. Start with next smallest
    size (since anything plus this will lead to the size needed)%
       size _ (index + 1) * zone[blkmin]; % words needed %
       FOR i _ (index - 1) DOWN UNTIL < 0 DO
          BEGIN
          IF &blk _ zone[zfrelst + i] THEN
             BEGIN %found free list%
             WHILE  &blk DO
                BEGIN
                % try this block and adjacent blocks up %
                   IF (&newblk _ combup(&blk, size - blk.sysblength,
                   &blk, &zone)) THEN
```

```
                breakup(&newblk, &zone, index);
                RETURN(TRUE);
                END;
            % try this block and adjacent blocks down %
              IF (combdown(&blk, size - blk.sysblength, &blk,
              &zone)) THEN
                BEGIN
                breakup(&blk, &zone, index);
                RETURN(TRUE);
                END;
            &blk _ blk[blksuc];
            END;
        END;
    END;
RETURN(FALSE);
END.

(breakup) PROCEDURE  %break up blk and put in index free list%     6E
  (blk REF,    %address of block to be subdivided%
  zone REF,    %address of free storage zone%
  index);  %free list index to be replenished%
  %-----------------%
  %This routine creates a number of blocks (by breaking up blk) of
  appropriate size for the free list indicated by index and links
  them into that free list in zone.%
  %-----------------%
  LOCAL
    length, %length of smaller blocks%
    base REF,   %temp, used in subdividing blk%
    i,      %loop control variable%
    r;      %temp, used to set up i%
  length _ (index+1)*zone[blkmin];
  DIV blk.sysblength/length, i, r;
  IF NOT r THEN BUMP DOWN i;
  UNTIL (i _ i -1) < 0 DO
    BEGIN
    &blk _ (&base _ &blk) + length;
    blk.sysblength _ (base.sysblength := length) - length;
    linkup(&base, &zone);
    END;
  &blk _ (&base _ &blk) + blk.sysblength;
  IF base.sysblength >= zone[blkmin] THEN
    linkup(&base, &zone)
  ELSE base.blkfree _ FALSE;
  RETURN;
  END.

(unlink) PROCEDURE  %unlink a block from its free list%           6F
  (blk REF,  %address of block to be unlinked from its free list%
  zone REF); %address of a free storage zone%
  %-------------%
  %this routine unlinks blk from the appropriate free list in zone%
  %-------------%
  LOCAL
    next REF, %address of next block in zone%
```

```
      prd REF;  %address of block pointed to by predecessor link%
   blk.blkfree _ FALSE;
   IF (&next _ &blk + blk.sysblength) < zone[zonend] THEN
      next.blkprevfree _ FALSE;
   IF &suc _ blk[blksuc] THEN
      suc[blkprd] _ blk[blkprd];
   IF &prd _ blk[blkprd] THEN
      prd[blksuc] _ blk[blksuc] := 0
   ELSE
      zone[zfrelst + blk.sysblength/zone[blkmin] - 1] _ blk[blksuc]
      := 0;
   blk[blkprd] _ 0;
   RETURN;
   END.

(linkup) PROCEDURE  %linkup blk into appropriate free list%          6G
   (blk REF,  %address of block to be linked into free list%
   zone REF); %address of a free storage zone%
   %--------------%
   %this routine links blk into the begining of the appropriate free
   list in zone%
   %--------------%
   LOCAL
      next REF,   %address of next block in zone%
      index;  %free list index for blk%
   blk.blkfree _ TRUE;
   index _ blk.sysblength/zone[blkmin]-1 + zfrelst;
   blk[blksuc] _ zone[index] := &blk;
   blk[blkprd] _ 0;
   [&blk+blk.sysblength-ADR-PER-WORD] _ blk.sysblength;
   IF (&next _ &blk+blk.sysblength) < zone[zonend] THEN
      next.blkprevfree _ TRUE;
   IF &blk _ blk[blksuc] THEN
      blk[blkprd] _ zone[index];
   RETURN;
   END.

(combup)   % CL: ;  combine up %
PROCEDURE (blk REF, wrdsneeded, strtblk REF, zone REF);              6H
   % Procedure description
      FUNCTION
         blk--REF- addr of next block to consider.
         wrdsneeded--words needed to fulfill this request.
         strtblk--REF- addr of first block considered.
      ARGUMENTS
         none
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL
```

```
        next REF,
        newblk REF;
    % combine up %
      IF blk.blkprevfree = 0 THEN RETURN(FALSE);
      &newblk _ &blk - [&blk - ADR-PER-WORD];
      IF newblk.sysblength >= wrdsneeded THEN
        BEGIN % combine all blocks down to strtblk %
        &next _ &newblk;
        UNTIL &next > &strtblk DO
          BEGIN
          unlink(&next, &zone);
          nlength _ next.sysblength := 0;
          complength _ complength + nlength;
          &next _ &next + nlength;
          END;
        newblk.sysblength _ complength;
        RETURN(&newblk);
        END
      ELSE RETURN(combup(&newblk, wrdsneeded-newblk.sysblength,
      &strtblk, &zone));
    % Return %
      RETURN;
    END.

(combdown)   % CL: ;  combine down %
PROCEDURE (blk REF, wrdsneeded, strtblk REF, zone REF);                6I
    % Procedure description
      FUNCTION
        none
      ARGUMENTS
        blk--REF- addr of newblk block to consider.
        wrdsneeded--words needed to fulfill this request.
        strtblk--REF- addr of first block considered.
      RESULTS
        proc-value
      NON-STANDARD CONTROL
        none
      GLOBALS
        none
      %
    % Declarations %
      LOCAL
        nsize, csize, prevblk REF,
        newblk REF;
    % combine down %
      csize _ blk.sysblength;
      &newblk _ &blk + csize;
      nsize _ newblk.sysblength;
      IF newblk.blkfree AND (nsize + csize <= &zone) THEN
        BEGIN % newblk block free %
        IF nsize >= wrdsneeded THEN
          BEGIN % unlink the blocks we have checked %
          unlink(&strtblk, &zone);
          UNTIL &newblk <= &strtblk DO
            BEGIN
```

```
                strtblk.sysblength _ strtblk.sysblength +
                newblk.sysblength;
                newblk.sysblength _ 0;
                &newblk _ &newblk - [&newblk - ADR-PER-WORD];
                END;
             RETURN(TRUE); % the block is strtblk %
             END
          ELSE % keep going down %
             RETURN(combdown(&newblk, wrdsneeded-newblk.sysblength,
             &strtblk, &zone));
          END;
    % Return %
       RETURN(FALSE);
    END.

%support routines to control storage allocator%
   %string handling%
      (getstring) PROCEDURE (length, zone);                          7A1
         %allocate a block in "zone" of  "length" characters  plus
         block header plus string header.  Initialize the string to
         empty.  Return string address.
         length       actual useable length of string desired
         zone         address of zone with free space in it
         SIGNALS if no more room in that block%
         %---------------%
         LOCAL blk REF;
         &blk _ getblk ((length + WORD/CHAR -1)/(WORD/CHAR)  + %string
         header% (2*ADDRESS)/WORD, zone);
         IF NOT &blk THEN
            err( smezos);
         IF CPU = 11 THEN &blk _ &blk + 4;
         blk.M _ length;
         *blk* _ NULL;
         RETURN (&blk);
         END.

      (freestring) PROCEDURE (string, zone);  %free this allocated
      string%                                                        7A2
         LOCAL outcome;
         IF CPU = 11 THEN string _ string - 4;
         RETURN[outcome](freeblk(string, zone : [outcome]));
         END.

FINISH
```

```
< NINE, TSPRT.NLS;12, >, 12-Jul-78 17:42 HGL ;;;;
FILE tsprt % <ARCSUBSYS>XL10 to <RELNINE>tsprt %% (arcsubsys,xl10,)
(RELNINE,tsprt.rel,) %
   (cprint) %***% PROCEDURE (da, stid1, stid2, type, coretflag % not
   currently used, but leave space. %, oldsw);                          1A
      %print the structure (type) addrssed by stid1,stid2;  oldsw is
      the address of an already open sequence work area or zero.  If
      zero, a sequence work area is opened by printg and closed;  if
      non-zero, the sequence is not closed by printg. %
      %-----------------%
      LOCAL
         wrapptr,  %loc of last wrap hit in current line%
         i, %control variable for loop%
         contsw,  %continue returning more statements%
         exitloop,  %to force out of loop but not quite yet%
         gaplength,  %for skipping gap between lines%
         char,   %current character%
         gapcol, %column of char preceding last invisible char%
         gapptr, %pointer to char preceding last invisible char%
         startp, %pointer to start of line character%
         prindt,  %number of columns to indent%
         lincnt, %count of lines printed for current statement%
         stid,   %current stid being printed%
         maxcol, %maximum number of columns for current line%
         stnrt,  %whether stmt numbers are to be printed on right%
         extraline,  %true if stmt number put on separate line%
         stnlength,  %save length of stmt no.%
         sw,     %address of sequence generator work area%
         top,    %counter for spacing at top of page%
         vspc1,  %viewspecs from sequence work area %
         vspc2,
         head,   %address of file header%
         cntrlstr, %address of non-printing char string%
         sig2, sig3, sig4,      %signal results%
         mkrptr; %marker table pointer%
      LOCAL STRING
         str[100],    %scratch string%
         stnsig[50];  %holds statement number or signature%
      LOCAL TEXT POINTER z1;
      LOCAL LIST tempargs[4], colist[1], crtnlist[1];
      REF oldsw, sw, da, mkrptr;

      IF stid1 = endfil THEN ABORT(eof,$"End of file.");
      da.dacrow _ 0;  %start a page for later pagination%
      IF da.davspec.vspagf THEN
         BEGIN
         top _ 3;
         UNTIL (top := top - vinc) <= 0 DO da.dacrow _ da.dacrow +
         vinc;
         END;
      pageno _ 0;
      contsw _ TRUE;
      *tlit* _ NULL;
      &sw _ IF &oldsw THEN &oldsw
         ELSE openseq (stid1, IF da.davspec.vsbrof THEN stid1 ELSE
```

```
      since stid1 and stid2 supposedly form a legitimate group! %,
      da.davspec, da.davspc2, da.dausqcod, da.dacacode);
   INVOKE (closesw);
   z1[0] _ IF stid1.stastr THEN da.dacsp ELSE stid1;

   UNTIL ((stid _ seggen (&sw)) = endfil) OR
     (type = stmtv AND sw.swcstid # stid1) OR (NOT contsw) DO
      BEGIN
      % get viewspecs from sequence work area. %
         vspc1 _ da.davspec _ sw.swvspec;
         vspc2 _ da.davspc2 _ sw.swvsp2;
      %set up work area s2work for READC%
         s2work _ stid;
         s2work[1] _
            IF vspc1.vsnamf OR stid.stastr THEN 1 %print name%
            ELSE fchtxt (stid); %skip past name%
         fechc1 (forward, $s2work);
      %markers (not allowed for now)%
         prmkrf _ 0;
         IF FALSE AND vspc1.vsmkrf AND NOT stid.stastr THEN
            BEGIN  %see if marker in statemnt%
            &mkrptr _ $mkrtb - $filhed +
               (head _ <FILMNP, filhdr>(stid.stfile));        1A19D2B1
            mkrend _ &mkrptr +
               [$mkrtbl - $filhed + head] * mkrl;
            mkrflg _ FALSE;
            FOR mkrptr UP mkrl UNTIL = mkrend DO
               IF mkrptr.mkpsid = stid.stpsid THEN
                  BEGIN
                  mkrflg _ TRUE;
                  mkrcnt _ mkrptr.mkccnt + 1;
                  EXIT;
                  END;
            END;
      prindt _         %indentation%
         CASE TRUE OF
            = vspc1.vsindf, =vspc1.vsrind:
               MAX (tpoffset, MIN (da.daind * (sw.swclvl-
                  (IF vspc1.vsrind AND (vspc1.vsbrof OR
                  vspc1.vsplxf) THEN sw.swslvl ELSE 1) %this has
                  value=1 except when relative indenting actually
                  takes effect%
               )+ tpoffset, da.damind, spacestr.M));
            ENDCASE tpoffset;
         % TPOFFSET is a global which the user can set via Execute
         Viewchange.  This allows the user to control the left
         margin of his print out. %
      *prbuf* _ *spacestr* [empty + 1 TO prindt];
      maxcol  _ da.damcol;   %max no. cols%
      stnrt   _ FALSE;
      IF vspc1.vsstnf AND stid.stpsid # origin AND
       NOT stid.stastr THEN
         IF NOT vspc1.vsstnr THEN
            BEGIN  %print statement number%
            IF vspc1.vssidf
```

```
                *prbuf* _ *prbuf*, '0, STRING( getsid(stid) )
              ELSE % display line numbers %
                fechnm (sw.swsvw, $prbuf);
           *prbuf* _ *prbuf*, SP;
           END
        ELSE   % statement numbers go on the right %
           BEGIN
           *stnsig* _ NULL;
           IF vspc1.vssidf
              THEN % display sid's %
                 *stnsig* _ '0, STRING( getsid(stid) )
              ELSE % display line numbers %
                 fechnm (sw.swsvw, $stnsig);
           stnlength _ stnsig.L;
           stnrt  _ TRUE;
           END;
        gapcol _ da.daccol _ prbuf.L * hinc;
     %line printing loop%
     FOR lincnt _ 0 UP UNTIL = vspc1.vstrnc DO
        BEGIN
        exitloop _ 0;
        gaplength _ 2;
        wrapptr _ 0;
        udpnwrap _ 1;
        IF inptrf THEN EXIT 2;
        gapptr _ startp _ prbuf.L;
        UNTIL (da.daccol >= maxcol + 1 OR exitloop) DO
           BEGIN
           IF da.daccol = udpwrapcol*udpnwrap THEN
              wrapptr _ prbuf.L;
           CASE char _ READC ($s2work) OF
              =SP:
                 BEGIN
                 gapptr     _ prbuf.L;
                 gapcol     _ da.daccol;
                 da.daccol _
                 da.daccol+putchr(&da,char,da.daccol,$prbuf);
                 END;
              =ENDCHR, =EOL, =CR:
                 BEGIN
                 gapptr     _ prbuf.L;
                 gapcol     _ da.daccol;
                 exitloop _ 1;  %to force out of loop on next
                 round%
                 END;
              =TAB:
                 BEGIN
                 gapptr     _ prbuf.L;
                 gapcol     _ da.daccol;
                 da.daccol _
                 da.daccol+putchr(&da,char,da.daccol,$prbuf);
                 IF (NOT wrapptr AND udpwrapcol AND da.daccol >
                 udpwrapcol*udpnwrap) THEN
                    BEGIN
                    *prbuf* _ *prbuf*[1 TO prbuf.L-1], *spacestr*[1
```

```
                    wrapptr _ gapptr;
                    END;
                 END;
              =CA, =LF, =CD, =BC, =BW, =$ascalt, IN [1B,32B], IN
              [34B,36B]: %an acceptable non-printing character%
                 BEGIN
                 cntrlstr _ <DSPGEN, npstrad>(char);%get np
                 representation (address)%
                 %see if non-printing character string will exceed
                 current line%
                 IF [cntrlstr].L + da.daccol >= maxcol THEN
                    BEGIN
                    s2work[1] _ s2work[1] - 1;
                    fechc1 (forward, $s2work);  %repeat last char
                    next time through%
                    REPEAT CASE (char _ EOL);
                    END
                 ELSE %fits on current line%
                    FOR i _ 1 UP UNTIL > [cntrlstr].L DO
                       BEGIN
                       char _ *[cntrlstr]*[i];
                       da.daccol _
                       da.daccol+putchr(&da,char,da.daccol,$prbuf);
                       IF da.daccol >= udpwrapcol*udpnwrap THEN
                          wrapptr _ prbuf.L -
                          (da.daccol-udpwrapcol*udpnwrap);
                       END;
                 END;
              ENDCASE
                 BEGIN
                 da.daccol _
                 da.daccol+putchr(&da,char,da.daccol,$prbuf);
                 END;
           IF wrapptr AND gapptr > wrapptr THEN
              BEGIN  %write out wrapped subline%
              IF NOT prtype(&da, wrapptr) THEN EXIT 3;
              *prbuf*[prindt+1 TO prbuf.L] _ *prbuf*[wrapptr+1 TO
              prbuf.L];
              gapptr _ gapptr - wrapptr;
              wrapptr _ 0;
              udpnwrap _ udpnwrap + 1;
              END;
        END;
     %by here, a line has been constructed. fix gapptr and
     gapcol if there were no invisibles or if the last invisible
     was the last character%
        IF startp = gapptr AND NOT exitloop THEN   %no
        invisibles in line%
           BEGIN
           %an extra character was collected in case it was a
           gap; it wasn't, so don't print it on this line%
           gapcol _ da.daccol - 1;
           IF wrapptr THEN
              DO  %write out wrapped subline(s)%
                 BEGIN
```

```
                    *prbuf*[prindt TO prbuf.L] _ *prbuf*[udpwrapcol
                    + 1 TO prbuf.L];
                    END
                UNTIL prbuf.L - 1 < udpwrapcol;
             wrapptr _ 0;
             gapptr _ prbuf.L - 1;
             gaplength _ 1;  %for skipping no gap chars%
             END;
          IF gapptr = prbuf.L - 1 THEN     %last char a gap (SP or
          TAB)%
             LOOP  %extra spaces or tabs at end of line placed on
             this line%
             CASE char _ READC($s2work) OF
                =SP, =TAB:  NULL;  %throw them away%
                =ENDCHR, =EOL, =CR:  EXIT LOOP;
                ENDCASE
                   BEGIN
                   %just store it for the next line of this
                   statement%
                   da.daccol _ da.daccol +
                   putchr(&da,char,da.daccol,$prbuf);
                   EXIT LOOP;
                   END;
       IF (char = ENDCHR OR lincnt = vspc1.vstrnc) AND stnrt AND
       da.daccol + stnsig.L + 2 <= maxcol THEN extraline _ FALSE
          % can cram statement number on right of this line (which
          is the last line of the statement), so don't print it
          out yet%
       ELSE
          BEGIN
          extraline _ TRUE;
          IF NOT prtype (&da, gapptr) THEN EXIT 2; %type the line%
          IF NOT (char = ENDCHR OR lincnt +1 = vspc1.vstrnc) THEN
             *prbuf* [prindt + 1 TO prbuf.L] _ *prbuf* [gapptr + 2
             TO prbuf.L]
          ELSE
             *prbuf* _ *spacestr*[empty + 1 TO prindt];
          gapcol _ da.daccol _ prbuf.L*hinc;
          END;
       z1[0] _ stid;
       IF char = ENDCHR THEN EXIT;
       END;

    % The statement is completed.%

    %statement numbers on right%
       IF stnrt THEN
          BEGIN
          (stnort):                                          1A19M1B
          IF extraline THEN
             BEGIN          %putting number on separate line%
             *prbuf* _ *spacestr*[1 TO maxcol - stnsig.L-2];
             END
          ELSE
             BEGIN
             *prbuf* _ *spacestr*[1 TO maxcol - da.daccol
```

```
                - stnsig.L -2];
                END;
            *prbuf* _ *prbuf*, SP,SP, *stnsig*;
            IF NOT (extraline AND vspc1.vsidtf AND NOT stid.stastr)
            THEN prtype(&da, prbuf.L);
            END
         ELSE extraline _ FALSE;
      %blank line and signature%
         IF vspc1.vsidtf AND NOT stid.stastr THEN
            BEGIN
            *stnsig* _ NULL;
            fechsig (stid, $stnsig);
            IF extraline AND stnlength + stnsig.L +3 > maxcol -
            prindt THEN
               BEGIN
               prtype(&da, prbuf.L);
               extraline _ FALSE;
               END;
            IF extraline THEN *prbuf*[maxcol - stnlength - stnsig.L
            -2 TO maxcol - stnlength -3] _ *stnsig*
            ELSE *prbuf* _
               *spacestr* [1 TO maxcol - stnsig.L], *stnsig*;
            prtype (&da, prbuf.L);
            END
         ELSE IF vspc1.vsblkf AND NOT extraline THEN prtype(&da, 0);
      END;
   % send any remaining buffer %
      IF tlit.L THEN prtype(&da, 0);
   IF NOT &oldsw THEN closeseq (&sw);
   DROP (closesw);
   RETURN;

   %Define the catchphrase, closesw%

   (closesw) CATCHPHRASE (:sig2, sig3, sig4);                    1A27
      CASE SIGNALTYPE OF
         = aborttype:
            BEGIN
            DISABLE (closesw);
            IF NOT &oldsw THEN closeseq(&sw);
            CONTINUE;
            END;
      ENDCASE CONTINUE;

   END.
(prtype) PROCEDURE (da, number);                                  1B
   %type "number" characters from prbuf; if number is less than the
   length of prbuf, prbuf will not be disturbed by the operation.%
   %-----------------%
   LOCAL length; %length of astring in prbuf%
   REF da;
   % ^D %
      IF inptrf THEN RETURN (FALSE);
   % look for page eject %
      IF da.davspec.vspagf AND da.dacrow >= da.damrow THEN
```

```
      da.dacrow _ da.dacrow + 1;
   IF number = 0 THEN
      BEGIN % Print out rest of buffer, if any %
      typseq(cttywindow, nocawait, clrwno, crba, belno, $tlit);
      *tlit* _ NULL;
      END
   ELSE
      IF number > 0 THEN
         BEGIN
         length _ prbuf.L := MIN(number, prbuf.L);
         IF (maxdpschar - tlit.L) <= prbuf.L THEN
            BEGIN
            typseq(cttywindow, nocawait, clrwno, crb, belno, $tlit);
            *tlit* _ NULL;
            END;
          IF tlit.L THEN
            BEGIN
            IF (tlit.L + prbuf.L + 1) >= MIN (maxdpschar, tlit.M)
            THEN
               BEGIN
               % if tlit will not take stuff to be appended to it,
               send the material out now. %
               typseq(cttywindow, nocawait, clrwno, crno, belno,
               $tlit);
               *tlit* _ NULL;
               END;
            *tlit* _ *tlit*, EOL, *prbuf*;
            END
          ELSE *tlit* _ *prbuf*;
          prbuf.L _ length; % restore length of buffer %
          END;
    IF inptrf %^0% THEN
       BEGIN
       RETURN (FALSE);
       END;
    RETURN (TRUE);
    END.

(newpag)  %page eject%                                              1C
PROCEDURE (da);
   % this procedure does a page eject for printing in TNLS. It takes
   into account the device being used, and tries to make the total
   page length equal to 11 inches, unless otherwise specified by the
   user%
   %-----------------%
   LOCAL toplines, bottom;
   LOCAL STRING pagnostr[10];  %for page number%
   REF da;
   % initialize work variables %
      bottom _ da.dabottom-da.datop-da.damrow;
      toplines _ 3;
      da.dacrow _ 0;
   % put in EOLs until page number line %
      IF (tlit.L + 2 + MAX( bottom-2,0) ) >= MIN (maxdpschar,
      tlit.M) THEN
```

```
        % if tlit will not take stuff to be appended to it, send
        the material out now. %
        typseq(cttywindow, nocawait, clrwno, crno, belno, $tlit);
        *tlit* _ NULL;
        END;
      *tlit* _ *tlit*, EOL;
      DO *tlit* _ *tlit*, EOL
      UNTIL (bottom _ bottom - 1) <= 2;
    % put out page number if room %
      IF bottom > 0 THEN
        BEGIN %Print page number%
        *pagnostr* _ "Page ",STRING(pageno _ pageno+1), EOL;
        IF (tlit.L + spacestr.L + pagnostr.L + 10) >= MIN
        (maxdpschar, tlit.M) THEN
          BEGIN
          % if tlit will not take stuff to be appended to it, send
          the material out now. %
          typseq(cttywindow, nocawait, clrwno, crno, belno,
          $tlit);
          *tlit* _ NULL;
          END;
        *tlit* _ *tlit*, *spacestr*[1 TO (da.damcol - pagnostr.L)],
        *pagnostr*, EOL,  "--------", EOL;
        END;
    % put out EOLs until top of next page %
      da.dacrow _ da.dacrow + toplines;
      IF (tlit.L + MAX( toplines, 0) ) >= MIN (maxdpschar, tlit.M)
      THEN
        BEGIN
        % if tlit will not take stuff to be appended to it, send
        the material out now. %
        typseq(cttywindow, nocawait, clrwno, crno, belno, $tlit);
        *tlit* _ NULL;
        END;
      UNTIL (toplines:= toplines - 1) <= 0
      DO *tlit* _ *tlit*, EOL;
      typseq(cttywindow, nocawait, clrwno, crb, belno, $tlit);
      *tlit* _ NULL;
    RETURN;
    END.

FINISH
```

(cmdindflt)	<nine, uoconst, 042>	EXT CONSTANT =255	4A3
(cmdmddflt)	<nine, uoconst, 043>	EXT CONSTANT =0	4A4
(curversio)	<nine, uoconst, 032>	EXT CONSTANT =9	3B1
(defcolmax)	<nine, uoconst, 011>	EXT CONSTANT =72	3A6
(defcurcon)	<nine, uoconst, 06>	EXT CONSTANT =7	3A1
(defdcolmax)	<nine, uoconst, 07>	EXT CONSTANT =72	3A2
(deffrrsize)	<nine, uoconst, 010>	EXT CONSTANT =10	3A5
(defindcnt)	<nine, uoconst, 015>	EXT CONSTANT =3	3A10
(deflinmax)	<nine, uoconst, 014>	EXT CONSTANT =60	3A9
(defoffset)	<nine, uoconst, 012>	EXT CONSTANT =0	3A7
(defpgsize)	<nine, uoconst, 013>	EXT CONSTANT =66	3A8
(defqcolmax)	<nine, uoconst, 016>	EXT CONSTANT =72	3A11
(defqindcnt)	<nine, uoconst, 020>	EXT CONSTANT =3	3A15
(defqlinmax)	<nine, uoconst, 019>	EXT CONSTANT =60	3A14
(defqoffset)	<nine, uoconst, 017>	EXT CONSTANT =0	3A12
(defqpgsize)	<nine, uoconst, 018>	EXT CONSTANT =66	3A13
(defqt1)	<nine, uoconst, 025>	EXT CONSTANT =776775773767B	
(defqt2)	<nine, uoconst, 026>	EXT CONSTANT =7577376775778	
(defqt3)	<nine, uoconst, 027>	EXT CONSTANT =376775773767B	
(defqt4)	<nine, uoconst, 028>	EXT CONSTANT =757737677577B	
(defsrrsize)	<nine, uoconst, 09>	EXT CONSTANT =10	3A4
(defth1)	<nine, uoconst, 021>	EXT CONSTANT =776775773767B	
(defth2)	<nine, uoconst, 022>	EXT CONSTANT =757737677577B	
(defth3)	<nine, uoconst, 023>	EXT CONSTANT =376775773767B	
(defth4)	<nine, uoconst, 024>	EXT CONSTANT =757737677577B	
(defvs1)	<nine, uoconst, 029>	EXT CONSTANT =300300007777B	
(defvs2)	<nine, uoconst, 030>	EXT CONSTANT =0	3A25
(defwrpcol)	<nine, uoconst, 08>	EXT CONSTANT =0	3A3
(dflttemp)	<nine, uoconst, 035>	EXT STRING	3A26
(dfprtdir)	<nine, uoconst, 036>	EXT STRING	3A27
(frstpage)	<nine, uoconst, 033>	EXT CONSTANT =554B	3B2
(hldindflt)	<nine, uoconst, 044>	EXT CONSTANT =5	4A5
(hldmddflt)	<nine, uoconst, 045>	EXT CONSTANT =0	4A6
(nsindflt)	<nine, uoconst, 040>	EXT CONSTANT =255	4A1
(nsmddflt)	<nine, uoconst, 041>	EXT CONSTANT =0	4A2
(prptdflt)	<nine, uoconst, 046>	EXT CONSTANT =2	4A7
(rcg2dflt)	<nine, uoconst, 048>	EXT CONSTANT =4	4A9
(rcgdflt)	<nine, uoconst, 047>	EXT CONSTANT =1	4A8
(uoext)	<nine, uoconst, 059>	EXT	3B3
(upanticipatory)	<nine, uoconst, 057>	EXT CONSTANT =0	4B8
(upconditional)	<nine, uoconst, 056>	EXT CONSTANT =2	4B7
(updemand)	<nine, uoconst, 058>	EXT CONSTANT =3	4B9
(upfixed)	<nine, uoconst, 054>	EXT CONSTANT =2	4B5
(upfull)	<nine, uoconst, 052>	EXT CONSTANT =0	4B3
(upoff)	<nine, uoconst, 055>	EXT CONSTANT =3	4B6
(uppartial)	<nine, uoconst, 053>	EXT CONSTANT =1	4B4
(upterse)	<nine, uoconst, 051>	EXT CONSTANT =1	4B2
(upverbose)	<nine, uoconst, 050>	EXT CONSTANT =0	4B1

```
< NINE, UOCONST.NLS;9, >, 8-Mar-78 15:10 LLG ;;;;
FILE uoconst % (arcsubsys,x110,) (arcsubsys,1109,) to
(reinine,uoconst.rel,) %
% DECLARATION of constants to be used with useroption subsystem %
% BE (useroptions) constants %
   % default values %
      (defcurcon) EXTERNAL CONSTANT = 7; % current context %          3A1
      (defdcolmax) EXTERNAL CONSTANT = 72; % display absolute right
      margin %                                                       3A2
      (defwrpcol) EXTERNAL CONSTANT = 0; % wrap around column (zero
      means no wraparound) %                                         3A3
      (defsrrsize) EXTERNAL CONSTANT = 10; % statement return ring size
      %                                                              3A4
      (deffrrsize) EXTERNAL CONSTANT = 10; % file return ring size % 3A5
      (defcolmax) EXTERNAL CONSTANT = 72; % printer right margin %   3A6
      (defoffset) EXTERNAL CONSTANT = 0; % printer left margin %     3A7
      (defpgsize) EXTERNAL CONSTANT = 66; % printer page size %      3A8
      (deflinmax) EXTERNAL CONSTANT = 60; % printer bottom margin %  3A9
      (defindcnt) EXTERNAL CONSTANT = 3; % printer indenting per level
      %                                                              3A10
      (defqcolmax) EXTERNAL CONSTANT = 72; % quickprint right margin %
                                                                     3A11
      (defqoffset) EXTERNAL CONSTANT = 0; % quickprint left margin %
                                                                     3A12
      (defqpgsize) EXTERNAL CONSTANT =66; % quickprint page size (in
      lines) %                                                       3A13
      (defqlinmax) EXTERNAL CONSTANT = 60; % quickprint bottom margin %
                                                                     3A14
      (defqindcnt) EXTERNAL CONSTANT = 3; % quickprint indenting per
      level %                                                        3A15
      (deftb1) EXTERNAL CONSTANT = 776775773767B; % printer tab
      settings %                                                     3A16
      (deftb2) EXTERNAL CONSTANT = 7577376775776;                    3A17
      (deftb3) EXTERNAL CONSTANT = 3767757737676;                    3A18
      (deftb4) EXTERNAL CONSTANT = 7577376775776;                    3A19
      (defqt1) EXTERNAL CONSTANT = 776775773767B; % quickprint tab
      settings %                                                     3A20
      (defqt2) EXTERNAL CONSTANT = 7577376775776;                    3A21
      (defqt3) EXTERNAL CONSTANT = 3767757737676;                    3A22
      (defqt4) EXTERNAL CONSTANT = 7577376775776;                    3A23
      (defvs1) EXTERNAL CONSTANT = 300300007777B; % default viewspecs %
                                                                     3A24
      (defvs2) EXTERNAL CONSTANT = 0;                                3A25
      (dflttemp) EXTERNAL STRING = "<nine,prog-templates,>";         3A26
      (dfprtdir) EXTERNAL STRING = "ARCPRINTER";  % default printer
      directory %                                                    3A27
   % general use %
      (curversio) EXTERNAL CONSTANT = 9;                             3B1
      (trstpage) EXTERNAL CONSTANT = 554B; % When user optins are first
      mapped in they will be placed in the user programming area, then
      copied into the global list and variables to which they refer.
      (See UODATA.) %                                                3B2
      (uoext) EXTERNAL _ $initsr;  %extension for USEROPTIONS file%  3B3
% FE (userprofile) constants %
   % userprofile defaults %
```

```
    (nsmddflt)  EXTERNAL CONSTANT = 0;     % verbose %                4A2
    (cmdlndflt) EXTERNAL CONSTANT = 255;                             4A3
    (cmdmddflt) EXTERNAL CONSTANT = 0;     % verbose %                4A4
    (hldlndflt) EXTERNAL CONSTANT = 5;                               4A5
    (hldmddflt) EXTERNAL CONSTANT = 0;     % verbose %                4A6
    (prptdflt)  EXTERNAL CONSTANT = 2;     % conditional %            4A7
    (rcgdflt)   EXTERNAL CONSTANT = 1;     % terse %                  4A8
    (rcg2dflt)  EXTERNAL CONSTANT = 4;     % terse %                  4A9
  % userprofile constants %
    (upverbose)       EXTERNAL CONSTANT = 0;                         4B1
    (upterse)         EXTERNAL CONSTANT = 1;                         4B2
    (upfull)          EXTERNAL CONSTANT = 0;                         4B3
    (uppartial)       EXTERNAL CONSTANT = 1;                         4B4
    (upfixed)         EXTERNAL CONSTANT = 2;                         4B5
    (upoff)           EXTERNAL CONSTANT = 3;                         4B6
    (upconditional)   EXTERNAL CONSTANT = 2;                         4B7
    (upanticipatory)  EXTERNAL CONSTANT = 0;                         4B8
    (updemand)        EXTERNAL CONSTANT = 3;                         4B9
FINISH of uodata
```

(bckspc)	<nine, uodata, 061>	EXT	2N1
(cntchr)	<nine, uodata, 0134>	EXT REF	3B
(colmax)	<nine, uodata, 026>	EXT	2I3
(dfnmd1)	<nine, uodata, 021>	EXT	2H1
(dfnmdr)	<nine, uodata, 022>	EXT	2H2
(enlistr)	<nine, uodata, 063>	EXT STRING	2P1
(entsubsystem)	<nine, uodata, 075>	EXT STRING	2Q1
(frrsize)	<nine, uodata, 012>	EXT	2E1
(indcnt)	<nine, uodata, 027>	EXT	2I4
(linmax)	<nine, uodata, 025>	EXT	2I2
(nolevadj)	<nine, uodata, 015>	EXT	2F1
(novspec)	<nine, uodata, 016>	EXT	2F2
(pad2)	<nine, uodata, 0139>	FIELD - 4	3F1
(pgsize)	<nine, uodata, 024>	EXT	2I1
(prfl)	<nine, uodata, 0138>	RECORD	3F
(profil)	<nine, uodata, 0135>	EXT REF	3C
(prtdir)	<nine, uodata, 073>	EXT STRING	2S1
(rjtchr)	<nine, uodata, 071>	EXT STRING	2N1
(rtjtab)	<nine, uodata, 041>	EXT	2L2
(spttab)	<nine, uodata, 040>	EXT	2L1
(srrsize)	<nine, uodata, 013>	EXT	2E2
(stdtab)	<nine, uodata, 029>	EXT	2I6
(stdvsp)	<nine, uodata, 038>	EXT	2K1
(stuplist)	<nine, uodata, 065>	EXT LIST	2R1
(subslist)	<nine, uodata, 066>	EXT LIST	2R2
(toolst)	<nine, uodata, 0133>	EXT REF	3A
(tpoffset)	<nine, uodata, 028>	EXT	2I5
(tslshchars)	<nine, uodata, 010>	EXT	2D1
(udpcolmax)	<nine, uodata, 019>	EXT	2G2
(udpwrapcol)	<nine, uodata, 018>	EXT	2G1
(uoend)	<nine, uodata, 069>	EXT	2C1
(uoversion)	<nine, uodata, 048>	EXT	2C1
(upbtstr)	<nine, uodata, 0136>	EXT	3D
(uprofil)	<nine, uodata, 0137>	EXT LIST	3E
(uqpcolmax)	<nine, uodata, 033>	EXT	2J3
(uqpindcnt)	<nine, uodata, 034>	EXT	2J4
(uqplinmax)	<nine, uodata, 032>	EXT	2J2
(uqppgsize)	<nine, uodata, 031>	EXT	2J1
(uqpstdtab)	<nine, uodata, 036>	EXT	2J6
(uqptpoffset)	<nine, uodata, 035>	EXT	2J5

```
< NINE, UODATA.NLS.7, >, 18-Oct-77 17:42 SKO ;;;;
FILE uodata % (ARCSUBSYS,XL10,) (ARCSUBSYS,L109,) (RELNINE,uodata.rel,)
%
% BE (useroptions) data %
   % this file contains the declarations for the NLS user-profile.
     default values for these declarations are contained in CONST.
     DO NOT CHANGE THE ORDER IN THIS FILE !!!!!!
     additions should be made only immediatelly preceeding uoend %

   % competibility check %
     (uoversion) EXTERNAL; % = 9 for NLS9 %                          2C1
   % current context %
     (tslshchars) EXTERNAL;     % no. of characters to either side of
     CM for /%                                                       2D1
   % jump ring %
     (frrsize) EXTERNAL;        %no. entries in file return ring%    2E1
     (srrsize) EXTERNAL;        %no. entries in statement return ring%
                                                                     2E2
   % level adjust and viewspec flags %
     (nolevadj) EXTERNAL;       % if TRUE don't ask for level adjust%
                                                                     2F1
     (novspec) EXTERNAL;        %if TRUE don't ask for viewspecs %   2F2
   % display %
     (udpwrapcol) EXTERNAL; % display wraparound %                   2G1
     (udpcolmax) EXTERNAL; %max no. of print columns for any display
     area %                                                          2G2
   % default name delimeters%
     (dfnmdl) EXTERNAL; %default left name delimeter%                2H1
     (dfnmdr) EXTERNAL; %default right name delimeter%               2H2
   % print options data %
     (pgsize) EXTERNAL;                 % number of lines per page% 2I1
     (linmax) EXTERNAL;                 % max no. of lines to print per
     page%                                                           2I2
     (colmax) EXTERNAL;                 % max no. of columns to print%
                                                                     2I3
     (indcnt) EXTERNAL;                 % indenting per level%      2I4
     (tpoffset) EXTERNAL;       % left margin adjustment for printg),
     number of
                          columns to offset left margin%             2I5
     (stdtab) EXTERNAL [4];     % standard tab settings %           2I6
   % quickprint %
     (uqppgsize) EXTERNAL; % number of lines per page %              2J1
     (uqplinmax) EXTERNAL; % max no. of lines to print per page %    2J2
     (uqpcolmax) EXTERNAL; % max no. of columns to print%            2J3
     (uqpindcnt) EXTERNAL; % indenting per level%                    2J4
     (uqptpoffset) EXTERNAL; % left margin adjustment--probably will
     never be used but put it in to match printoptions%              2J5
     (uqpstdtab) EXTERNAL [4]; % standard tab settings%              2J6
   % viewspecs %
     (stdvsp) EXTERNAL [2];     % standard viewspecs %              2K1
   % tab spacing.  See printoptions for tab stops %
     (spftab) EXTERNAL; % if true input spaces instead of tab chr % 2L1
     (rtjtab) EXTERNAL;  % if true right justify tabbed material %  2L2
   % backspace control %
     (bckspc) EXTERNAL; %if true backspace by moving carriage
     backwards %                                                     2M1
```

```
   % characters to terminate right justification mode %
      (rjtchr) EXTERNAL STRING[5];                                    2N1
   % end of useroption data %
      (uoend) EXTERNAL;                                               2O1
   % external name link file address string %
      (enlfstr) EXTERNAL STRING[40];                                  2P1
   % entry subsyste name %
      (entsubsystem) EXTERNAL STRING[40];                             2Q1
   % startup and subsystems LISTs %
      (stuplist) EXTERNAL LIST [1]; % the element being a link string %
                                                                      2R1
      (subslist) EXTERNAL LIST [4]; %list of strings of subsystem
      names%                                                          2R2
   % default directory for output (spooler dir) %
      (prtdir) EXTERNAL STRING = "ARCPRINTER"; % [39]; %              2S1
% BE (userprofile) data %
   (toolst) EXTERNAL REF;  % list of default tools -- NOT IMPLEMENTED %
                                                                      3A
   (cntchr) EXTERNAL REF;   % list of control characters %            3B
   (profil) EXTERNAL REF;   % pointer to upbtstr %                    3C
   (upbtstr) EXTERNAL _ (26, 0);  % bitstr containing herald prompt etc
   %                                                                  3D
   (uprofil) EXTERNAL LIST [5];  % entire userprofile data structure %
                                                                      3E
   (prtl) RECORD  % definition of herald, prompt, etc %               3F
      pad2[4] %to 36 bits%, pad1[6] %to 32 bits%, rcg2[2], rcg[2],
      cmdmd[1], cmdlen[4], nsmd[1], nslen[8], prpt[2], hldmd[1],
      hldlen[5];

FINISH of uodata
```

(addsubs)	<nine, uonls, 0436>	PROCEDURE	3F
(away)	<nine, uonls, 0549>	CONSTANT =203	2C
(decuopt)	<nine, uonls, 0142>	PROCEDURE	6A
(in)	<nine, uonls, 0547>	CONSTANT =201	2A
(nlsupinit)	<nine, uonls, 0558>	PROCEDURE	4B
(out)	<nine, uonls, 0548>	CONSTANT =202	2B
(rstupt)	<nine, uonls, 0576>	PROCEDURE	4C
(setptr)	<nine, uonls, 0590>	PROCEDURE	4D
(uoinit)	<nine, uonls, 09>	PROCEDURE	3A
(uoisize)	<nine, uonls, 0550>	CONSTANT =20	2D
(uopen)	<nine, uonls, 0506>	PROCEDURE	3C
(uoproginit)	<nine, uonls, 0699>	PROCEDURE	3B
(uorstall)	<nine, uonls, 047>	PROCEDURE	3D
(uorstdflt)	<nine, uonls, 084>	PROCEDURE	3E
(upinit)	<nine, uonls, 0600>	PROCEDURE	4E
(upopen)	<nine, uonls, 0640>	PROCEDURE	4F
(uprsetall)	<nine, uonls, 0656>	PROCEDURE	4G

```
< NINE, UONLS.NLS;13, >, 6-Apr-78 09:31 LLG ;;;;
FILE uonls % (arcsubsys,x110,) (arcsubsys,1109,) to
(reinine,uonls.rel,) %
% DECLARATIONS %
   (in) CONSTANT = 201;                                                    2A
   (out) CONSTANT = 202;                                                   2B
   (away) CONSTANT = 203;                                                  2C
   (uolsize) CONSTANT = 20;  %length for useroption data list%             2D
% BE (useroptions) parameter manipulation code %
   (uoinit) % initialize useroption data %
   PROCEDURE;                                                              3A
      %
         FUNCTION:
            read useroption file and initialize data in core
         ARGUMENTS
            none
         RESULTS
            TRUE always
         NON-STANDARD CONTROL
            Catches aborts due to loading errors and reports them to
            the user via dismes and TERMINATES abort.
      %
      LOCAL
         pcpport,
         errstr REF,
         uojfn, numpage, uolist REF;
      % check if file exists %
         IF NOT (uojfn _ uopen(1B11 %old only%)) THEN % file doesn't
         exist %
            BEGIN
            % Use default useroptions: don't need to create a new file
            %
            uorstall();
            END
         ELSE
            BEGIN
            % read data from file %
               numpage _ filsize(uojfn);
               mapout( frstpage, frstpage + numpage + 1);
               uomap( uojfn, in, numpage, frstpage );
            % transfer data %
               blkxfr( frstpage*1000B, $uoversion, $uoend-$uoversion +
               1);
               IF uoversion # curversion THEN %incompatible data%
                  BEGIN
                     %dismes( 1, $"incompatible useroption data -
                     default used");%
                     uorstall();
                  END
               ELSE % decode the data %
                  BEGIN
                  OPENPORT rlist((frstpage+1)*1000B, $dspblk :
                  [pcpport]);
                  &uolist _ decuopt(pcpport).RH;  % addr in RH of list
                  descriptor %
                  *uolfstr*  *F FLEM #uolist#F11 ]*;
```

```
                #stuplist#[1] _ MOVE #uolist#[2];
                #subslist# _ MOVE #[ ELEM #uolist#[3] ]#;
                *tmpfile* _ *[ELEM #uolist# [4] ]*;  % programmer's
                template file %
                % These elements were added after NLS9 was on the air
                and users already had useroption data files.
                Therefore, the size of the list must be used to
                indicate whether or not the usertoptin data file
                includes these elements %
                   IF uolist.L= uolsize THEN
                      BEGIN
                      *prtdir* _
                         *[ ELEM #uolist#[5] ]*;  %printer directory
                         %
                      *entsubsystem* _
                         *[ ELEM #uolist#[6] ]*;  %entry subsystem %
                      END;
                #uolist# _; %clean up %
                END;
          % restore core status %
             IF tops20flag THEN uomap( uojfn, away, numpage,
             frstpage);
             finjfn( uojfn );
             mapin( frstpage, frstpage + numpage + 1);
          END;
    RETURN;
    END.
(uoproginit)   % CL: ;  load default subsystems and programs %
PROCEDURE;                                                          3B
    % Procedure description
       FUNCTION
          none
       ARGUMENTS
          none
       RESULTS
          proc-value
       NON-STANDARD CONTROL
          none
       GLOBALS
          none
       %
    % Declarations %
       LOCAL errstr REF;
    % load default subsystems and programs %
       INVOKE (catbadload, dropcat);
       addsubs($subslist);
       (dropcat):                                                   3B3C
       DROP (catbadload);
    % Return %
       RETURN;
    % catchphrases %
       (catbadload) CATCHPHRASE(:&errstr);                          3B5A
          BEGIN
          CASE SIGNALTYPE OF
             = aborttype :
```

```
                    = badload:
                       BEGIN  %could not load some default subsystem%
                       dismes(1,$errstr);
                       TERMINATE;
                       END;
                    ENDCASE;
                ENDCASE;
            CONTINUE;
            END;
    END.

(uopen) % open useroption file %
PROCEDURE ( flags );                                                    3C
    %
        FUNCTION:
            opens useroption file for readwrite according to flags
        ARGUMENTS:
            flags - integer - flags word for GTJFN JSYS
        PROCEDURE VALUE
            The JFN if successful FALSE otherwise
    %
    LOCAL ljfn;
    LOCAL STRING filnam[100];
    % create the filename %
        *filnam* _ '<, *userstr*, '>, "USEROPTION.", *[uoext]*,
        fvrdchar, "1;P727200", 0;
    % try to get the jfn %
        IF NOT ( ljfn _ sgtjfn( flags.LH .V 1B6, $filnam, $lit) )
            THEN RETURN( FALSE );
    % try to open the file %
        IF NOT sysopen(ljfn, readwrite, bintyp, $filnam) THEN
            BEGIN % unsuccessful open %
                reljfn( ljfn );
                RETURN( FALSE );
            END;
    RETURN( ljfn );
    END.
(uorstall) % reset entire useroptions data %
PROCEDURE;                                                              3D
    %
        FUNCTION:
            reset entire useroptions data
        PROCEDURE VALUE
            TRUE always
    %
    % procedure body %
        tslshchars _ defcurcon;
        udpcolmax _ defdcolmax;
        udpwrapcol _ defwrpcol;
        (dfnmdl, dfnmdr) _ ( 0,0 );
        srrsize _ defsrrsize;
        frrsize _ deffrrsize;
        colmax _defcolmax;
        tpoffset _ defoffset;
        pgsize _ defpgsize;
        linmax _ deflinmax;
```

```
      indcnt _ defindcnt;
      (stdtab, stdtab[1], stdtab[2], stdtab[3]) _ (deftb1, deftb2,
      deftb3, deftb4);
      uqpcolmax _ defqcolmax;
      uqptpoffset _ defqoffset;
      uqppgsize _ defqpgsize;
      uqplinmax _ defqlinmax;
      uqpindcnt _ defqindcnt;
      (uqpstdtab, uqpstdtab[1], uqpstdtab[2], uqpstdtab[3]) _
      (defqt1, defqt2, defqt3, defqt4);
      (stdvsp, stdvsp[1], novspec) _ (defvs1, defvs2, FALSE);
      nolevadj _ FALSE;
      spftab _ FALSE;
      rtjtab _ FALSE;
      bckspc _ FALSE;
      rjtchr _ 5B6; %empty string of .M=5 %
      uoversion _ curversion;
      uorstdflt();
      enlfstr.L _ 0;
      #stuplist#[1] _ ""; %empty string %
      *tmpfile* _ *dflttemp*;  % programmer's template file %
      *prtdir* _ *dfprtdir*;  % printer directory %
   RETURN;
   END.
(uorstdflt) % reset default subsystems %
PROCEDURE;                                                          3E
   %
      FUNCTION:
         assign default subsystems to users list
      PROCEDURE VALUE
         TRUE always
   %
   #subslist# _ "BASE", "PROGRAMS";
   RETURN;
   END.
(addsubs)   % CL: ;  load default subsystems  %
PROCEDURE (slist REF);                                              3F
   % Procedure description
      FUNCTION
         Load any default subsystems and programs that are not
         built-in in NLS.  Subslist, read as part of the
         useroptions, is used as a source of links.
      ARGUMENTS
         slist--REF-address of list holding links of oldfilenames.
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         ABORT with error code 'badload' and message "Cannot Load
         Subsytem" message if cannot load some subsystem.
      GLOBALS
         none
      %
   % Declarations %
      LOCAL
         adstr[40],
         i, j, bltinsubs REF;
```

```
      LOCAL TEXT POINTER tp1;
      LOCAL STRING
         subname[78], errstr[1999];
   % don't load builtin subs (BASE & those in nlssubs) %
      FOR i _ 1 UP UNTIL > slist.L DO
         BEGIN
         *subname* _ *[ ELEM #slist# [i] ]*;
         IF *subname* = *pkgname* THEN REPEAT LOOP;
         j _ 0;
         WHILE &bltinsubs _ nlssubs[j] DO
            BEGIN
            IF *subname* = *bltinsubs* THEN REPEAT LOOP 2;
            j _ j + 2;
            END;
         *subname* _ '<, *subname*, ",>" ;  % make it a link %
         FIND SF(*subname*) ^tp1;
         lnkprs($tp1, $adstr);
         IF cldprog($adstr, FALSE %don't dismes%, $errstr) THEN %
         bad loading %
            BEGIN
               *errstr* _ *errstr*, "     Cannot Load ", *subname*;
               ABORT(badload, $errstr);
            END;
         END;
   % Return %
      RETURN;
   END.

% FE (userprofile) parameter manipulation code %
   % DECLARATIONS %
      % userprofile data structures etc %
         REF toolst, cntchr, profil;
   (nlsupinit)   % userprofile initialization within NLS %
   PROCEDURE( rtnlist REF );                                          4B
      % Procedure description
         FUNCTION
            read userprofile and pass to fe
         ARGUMENTS
            none
         RESULTS
            the userprofile in list form
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      upinit(frstpage,  3);
      #rtnlist# _ LIST( MOVE #uprofil# );
      % Return %
         RETURN;
      END.

   (rstupf) % create an empty user profile %
   PROCEDURE;                                                         4C
      %
```

```
        create an empty user profile list to be filled via
        uprsetall
   %
   #uprofil# _
      USE makedesc( ubitst, $upbtstr, FALSE), %feedback herald etc %
      "", %startup string %
      LIST(NULL), %default tools%
      LIST(NULL), %cont character list%
      USE makedesc(uindex, curversion, FALSE)
      ;
   RETURN;
   END.

(setptr) % set main structure pointers %
PROCEDURE;                                                          4D
   %
      FUNCTION:
         set the main list pointers
   %
   &profil _ ELEM #uprofil#[1] + 1 ;
   &toolst _ ELEM #uprofil#[3];
   &cntchr _ ELEM #uprofil#[4];
   RETURN;
   END.

(upinit) % initialize userprofile data %
PROCEDURE(fpage, numpage);                                          4E
   %
      FUNCTION:
         read data file and initialize data
      ARGUMENTS
         fpage - first page for data mapping
         numpage - number of pages available for mapping
      PROCEDURE VALUE
         TRUE always
   %
   LOCAL pcpport, upjfn, pt REF;
   % check if file exists and read data %
      IF NOT (upjfn _ upopen(1B11 %old only%)) THEN % file doesn't
      exist %
         BEGIN
         % Use default userprofile: don't need to create a new file
         %
         rstupf(); setptr();  uprsetall();
         END
      ELSE % file exists %
         BEGIN
         % read data from file %
            IF filsize( upjfn ) > numpage THEN
               err($"not enough room for userprofile - aborted!");
            numpage _ filsize(upjfn);
            uomap( upjfn, in, numpage, fpage );
         % decode the data %
            OPENPORT rlist(fpage*1000B, $dspblk : [pcpport]);
            &pt _ decuopt(pcpport).RH;  % addr in RH of list
```

```
            #uprofil# _ MOVE #pt# ;
            IF ELEM#uprofil#[5] # curversion THEN
               BEGIN
                  dismes( 1, $"incompatible userprofile - default
                  created");
                  rstupf(); setptr();  uprsetall();
               END
            ELSE setptr();
         % restore core status %
            uomap( upjfn, away, numpage, fpage);
            sysclose( upjfn, $lit );
         END;
   RETURN;
   END.

(upopen) % open userprofile file %
PROCEDURE (flags );                                                    4F
   %
      FUNCTION:
         Gets a jfn (according to flags) and opens the userprofile
         fil.
         Returns the jfn.
      ARGUMENTS:
         flags - integer - flag word for gtjfn JSYS.
   %
   LOCAL ljfn;
   LOCAL STRING upflnm[199];
   *upflnm* _ "<, *userstr*, ">USERPROFIL.", *initsr*, fvrdchar,
   "1;P727200", 0;
   IF (ljfn _ sgtjfn(flags, $upflnm, $lit)) THEN
      IF sysopen(ljfn, rwthawed, bintyp, $lit) THEN RETURN(ljfn)
         ELSE reljfn(ljfn);
      RETURN(FALSE);
   END.

(uprsetall) % resets entire userprofile %
PROCEDURE;                                                             4G
   %
      FUNCTION:
         resets all the parameters of the USERPROFILE
   %
   #toolst# _ NULL; % entry tool and defaults %
   profil.nsmd _ nsmddflt;
   profil.nslen _ nslndflt;
   profil.cmdlen _ cmdlndflt;
   profil.cmdmd _ cmdmddflt;
   profil.hldmd _ hldmddflt;
   profil.hldlen _ hldlndflt;
   profil.rcg _ rcgdflt;
   profil.rcg2 _ rcg2dflt;
   profil.prpt _ prptdflt;
   #uprofil#[2] _ ""; % startup branch = NULL %
   #cntchr# _ NULL ; % default control characters %
   RETURN;
   END.
```

```
% useroption and NLS interface %
% Decoding data procedures %
   (decuopt)   % LB: ;  decode data structure from PCPB-8 to L10 %
   PROCEDURE (pcpport % => led, type %);                                    6A
      % Procedure description
         FUNCTION
            convert one element in a PCPB-8 format data structure into
            an L10 List Element
         ARGUMENTS
            pcpport:  RLIST port
         RESULTS
            led: a global data structure will be allocated and a list
            element descriptor for it is returned
            type:  L10 type
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      % Declarations %
         LOCAL newds REF, type, utype, i, led REF, value, wsize, ladr
         REF;
      %decode structure type%
         &led _ 0;
         CASE type _ PCALL [pcpport] (pcpany: &newds) OF
            = pcpempty:
               BEGIN
               utype _ unull;
               &led _ makedesc(utype,0,FALSE);
               END;
            = pcpboolean, = pcpindex:  %newds has value%
               BEGIN
               utype _ IF type = pcpboolean THEN uboole ELSE uindex;
               &led _ makedesc(utype,&newds,FALSE);
               END;
            = pcpinteger:  %newds has address of word containing
            integer%
               BEGIN
               value _ newds;  %get value%
               IF value IN[0,777777B] THEN &led _
               makedesc(uinteg,value,FALSE) ELSE
                  BEGIN
                  &led _ alobl k(1);
                  led _ value;
                  &led _ makedesc(uinteg,&led,TRUE);
                  END;
               END;
            = pcpcharstr:  %newds has address of string%
               BEGIN
               &led _ rtnstring(&newds);  %copy string and make
               descriptor%
               freeblk(&newds, $dspblk);  %RLIST uses dspblk%
               END;
            = pcpbitst:  %newds has address of bit string%
               BEGIN
               % NEED TO CONVERT TO 36 BITS/WORD FROM 32 BITS/WORD %
```

```
            wsize _ (newds + 35) / 36;  %calculate no. of words%
            &led _ aloblk(wsize+1);
            led _ newds;
            blkxfr(&newds + 1,&led + 1,wsize);  %copy bitstring%
            &led _ makedesc(ubitst,&led,TRUE);
            END;
         = pcplist:  %newds has number of elements in list%
            BEGIN
            &ladr _ getlst(&newds);  %allocate storage for list%
            &led _ makedesc(ulist,&ladr,TRUE);
            FOR i _ 1 UP UNTIL > &newds DO  %convert each element%
               #ladr#[i] _ USE decuopt(pcpport :type);
            END;
         ENDCASE ABORT(ermptype, $"Illegal PCP data type", type);
   % Return %
      RETURN(&led, type);
   END.
```

FINISH of uonls

(away)	<nine, uoosi, 07>	CONSTANT =203	3C
(finjin)	<nine, uoosi, 032>	PROCEDURE	4
(in)	<nine, uoosi, 05>	CONSTANT =201	3A
(out)	<nine, uoosi, 06>	CONSTANT =202	3B
(uomap)	<nine, uoosi, 054>	PROCEDURE	5

```
< NINE, UOOSI.NLS;5, >, 13-Apr-78 15:53 LLG ;;;;
FILE uoosi %(arcsubsys,x110,) to (relnine,uoosi.rel,) %
ALLOW!
% DECLARATIONS %
   (in) CONSTANT = 201;                                                    3A
   (out) CONSTANT = 202;                                                   3B
   (away) CONSTANT = 203;                                                  3C
(rinjfn) % close a file and release a jfn %
PROCEDURE ( ljfn );                                                         4
   %
      FUNCTION:
         close a file and release a jfn
      ARGUMENTS:
         jfn - integer - file-handle to close
      PROCEDURE VALUE
         TRUE if successfully closed FALSE otherwise
   %
   IF SKIP !closf( ljfn.RH ) THEN RETURN(TRUE) ELSE RETURN(FALSE);
   END.
(uomap) % map pages between core and file %
PROCEDURE (ljfn, type, numpage, startpage) ;                                5
   %
      FUNCTION:
         maps pages between file (given by ljfn) and core image (
         starting at frstpage ) according to type.
      ARGUMENTS:
         ljfn - integer - file handle
         type - integer:
            =in: map pages from file to core
            =out: map pages from core to file
            =away:  remove pages from core image
         numpage - integer - number of pages to mAp
      PROCEDURE VALUE
         TRUE always
   %
   LOCAL r1, r2, i;
   % set the accumulators value %
      CASE type OF
         = in:
            BEGIN
               r1.LH _ ljfn ;
               r1.RH _ 1; % pages in file start at # 1 %
               r2.LH _ 485; % this fork %
               r2.RH _ startpage; % pages in core start at frstpage %
            END;
         = out:
            BEGIN
               r1.LH _ 485; % this fork %
               r1.RH _ startpage; % pages in core start at frstpage %
               r2.LH _ ljfn ;
               r2.RH _ 1; % pages in file start at # 1 %
            END;
         = away:
            BEGIN
               r2.LH _ 485; % this fork %
               r2.RH _ startpage; % pages in core start at frstpage %
```

```
            END;
      ENDCASE;
   % make the mapping %
      FOR i _ 1 UP UNTIL > numpage DO %map %
         BEGIN
            IF type = away THEN r1 _ -1;
            !pmap(r1, r2, 14B10 %readwrite%) ;
            BUMP r1; BUMP r2;
         END;
   RETURN;
   END.
FINISH of uoosi
```

```
< NINE, UTILITY.NLS;9, >, 8-May-78 16:02 BLP ;;;;( NLS, UTILTY.NLS;21,
), 23-MAY-74 12:51 HGL ;
FILE utility % <ARCSUBSYS>XL10  <RELNINE>utility %  % (arcsubsys,xL10,)
(RELNINE,utility.rel,) %
ALLOW!
% *** RECORD DEFINITIONS MOVED TO (CONST,) *** %
% Declarations%
   % Flag setting obsolete JSYS values that are now implemented via
   call on 'flagut' %
      (tstfg) EXTERNAL ADDRESS = 432B;                                  4A1
      (setfg) EXTERNAL ADDRESS = 433B;                                  4A2
      (rstfg) EXTERNAL ADDRESS = 434B;                                  4A3
   REGISTER
      p=7, wa=8, rp=9;
   EXTERNAL ctrlc;
   %fields used to save sdb initials%
      (cint4) EXTERNAL FIELD = [(rp), 5:0];                             4D1
      (cint3) EXTERNAL FIELD = [(rp), 5:5];                             4D2
      (cint2) EXTERNAL FIELD = [(rp), 5:10];                            4D3
      (cint1) EXTERNAL FIELD = [(rp), 5:15];                            4D4
      (oldint) EXTERNAL FIELD = [(rp), 21:15];                          4D5
% sdb destruction%
% handle control-c properly for remote displays%
   (ctcproc) PROCEDURE;                                                  6A
      (ctrlc);                                                          6A1
      % save the accumulators %
      svac1 _ R1; R1 _ $svacs; !BLT R1, svacse;
      S _ S + 40000040B;
      doctrlc();
      !HRLZI R1, svacs;
      !BLT R1, 17B;
      R1 _ svac1;
      !JSYS debrk;
      END.
   (doctrlc) PROCEDURE;                                                  6B
      LOCAL STRING  send [5];
      % Commented out until NLS9 supports IMLACs
      IF nldevice = imlac0 OR nldevice = imlac1  THEN
         BEGIN
         %%do "tsndn" jsys - turn on tty simulation%%
         *send* _ begmsg, 1+remfudge, remtsn;
         !sout(dspjfn,chbmty+$send,-send.L);
         END;
      %
      IF (tenex >= 13200) AND (nldevice = devlproc) THEN
         BEGIN
         *send* _ lpesc, lpnocoor;
         !sout(dspjfn,chbmty+$send,-send.L);
         END;
      !haltf(); %stop, can continue from here%
      % if it gets back here then he typed continue; turn off tty
      simulation and continue like nothing happened%
      %Commented out until NLS9 supports IMLACs
      IF nldevice = imlac0 OR nldevice = imlac1  THEN
         BEGIN
         %%do "tsndf" jsys - turn off tty simulation%%
```

```
        *send* _ begmsg, 1+remfudge, remtsf;
        !sout(dspjfn,chbmty+$send,-send.L);
        END;
    %
    RETURN;
    END.
% trap control characters (^c, rubout, ^o, ^S)%
 (trapcc) PROCEDURE;                                                    7A
    % trap control characters that could rip control away (like
    control c).  This rountine can be called many times.  Calls to it
    should be pairwise matched with calls on notrapcc.  only the
    first call will actually do anything. ditto for the matching call
    on notrapcc. %
    LOCAL capsav;
    capsav _ 0;
    IF trpcnt < 0 THEN trpcnt _ 0;
    IF (trpcnt := trpcnt+1) = 0 THEN
        BEGIN
        ccignore _ 50;
        savchntab[1] _ chntab[1] := $traps .V 1B6;
        savchntab[2] _ chntab[2] := $trapc .V 1B6;
        savchntab[3] _ chntab[3] := $trapo .V 1B6;
        !rpcap(4B5);
        capsav _ R3;
        !epcap(4B5, R2, R3 .V 4B11); %Permits process to assign ^C for
        pseudo interrupt%
        R1.LH _ 3;
        R1.RH _ 2;
        !JSYS ati;
        R1 _ 4B5;
        R2 _ 1B11; %activate ^c%
        !JSYS aic;
        END;
    RETURN(capsav);
    END.

 (notrapcc) PROCEDURE(capsav);                                          7B
    % undoes the effect of the last call on trapcc.%
    IF trpcnt <= 0 THEN trpcnt _ 1;
    IF (trpcnt _ trpcnt-1) = 0 THEN
        BEGIN
        chntab[1] _ savchntab[1];
        chntab[2] _ savchntab[2];
        chntab[3] _ savchntab[3];
        ccignore _ 0;
        %De-activate ^C channel%
            R1 _ 4B5;
            !JSYS rcm; %read channel mask--134%
            IF R1 .A 1B11 THEN
                BEGIN %de-activate it%
                R1 _ 4B5;
                R2 _ 1B11;
                !JSYS dic;
                R1 _ 3;
                !JSYS dti;
                END;
```

```
        % Disable ^C intercept capability by enabling capabilities
        previously in effect. %
           !epcap(485, R2, capsav);
        END;
     RETURN;
     END.

% trap illegal instructions %
% error handling routines %
   (werr) PROCEDURE (errno);                                          9A
      LOCAL errmes;
      errmes _ CASE errno OF
         = 2: $"S-stack overflow";
         = 3: $"Undetermined stack overflow";
         = 4: $"Bad file";
         = 5: $"Spec stack error";
         = 6: $"Pattern stack overflow";
         >= errmsbase: errno; %address of A-string%
         ENDCASE $"Error";
      ABORT(werrsig, errmes);
      END.

   (err) PROCEDURE (errno);                                           9B
      % errno is either the address of a string or a standard error
      number.  ERR generates a signal ERRSIG. %
      LOCAL errmes;
      %Save of address of call, and value of M%
         ermark _ M;
         ercall _ [M .A 18M] .A 18M - 1;
      errmes _ CASE errno OF
         = copyflag: $"File copy fails";
         = 2: $"Open scratch fails";
         = 3: $"Cannot load program";
         = 4: $"I/O Error";
         = 5: $"System error: String Variable Overflowed";
         = 6: $"Bad file block";
         = notyet: $"Not implemented";
         >= errmsbase: errno; %address of A-string%
         ENDCASE $"Error";
      ABORT(errsig, errmes);
      END.

% bit table routines%
   %assign a bit from specified table%
      (assnbit) PROCEDURE (table,length);                             10A1
         LOCAL end,count;
         end _ table + length;
         count _ 0;
         UNTIL table >= end DO BEGIN
            R1 _ [table];
            !JFFO 1,assnb0; %find a free bit%
            BUMP table;
            count _ count + 36
            END;
         %if it exits to here then no bits available!!%
         err($"assnbit: no more bits to assign");
```

```
      (assnb0): %found free bit set it assigned and return bit no.%
                                                                    10A1G
         R4 _ 400000000000B;
         R3 _ -R2;
         !LSH 4,(3); %justify the bit for change%
         !TDZ 1,4; %turn it off, thereby setting it!%
         R2 _ R2 + count +1; % add words count plus one (nubers
         start at 1!!)%
         [table] _ R1;
         RETURN(R2);
         END.

%set a specific bit in specified table%
   (setbit) PROCEDURE (table,bit,length);                           10B1
      table _ table + bit / 36;
      R3 _ -((bit MOD 36)-1);
      R1 _ [table];
      R4 _ 400000000000B;
      !LSH 4,(3); %justify the bit for change%
      !TDZ 1,4; %turn it off, thereby setting it!%
      [table] _ R1;
      RETURN;
      END.
%get the bit no. of the next assigned bit%
   (nxtbit) PROCEDURE (table,bit,length);                           10C1
      LOCAL end, count, quot, rem;
      end _ table + length;
      IF bit = 0 THEN
         BEGIN
         count _ 0;
         R1 _ [table];
         END
      ELSE
         BEGIN
         DIV bit / 36, quot, rem;
         table _ table + quot;
         count _ quot * 36;
         R1 _ -rem;
         !HRLZI 2,400000B;   %mask 1st word%
         !ASH 2,0(1);
         R1 _ [table];
         !IOR 1,2;
         END;
      DO
         BEGIN
         !SETCA 1,0;
         !JFFO 1,nxtbt0; %find an assigned bit%
         BUMP table;
         R1 _ [table];
         count _ count + 36;
         END
         UNTIL table >= end;
      %if it exits to here then no bits assigned!!%
         RETURN(FALSE);
      (nxtbt0): %found assigned bit, return bit no.%                10C1G
         R2 _ R2 + count +1; % add words count plus one (numbers
```

```
            start at 1!!)%
            RETURN(R2);
            END.

   %deassign a bit in specified table%
      (dassnbit) PROCEDURE (table,bit,length);                    10D1
         IF bit = -1 THEN %deassign all%
            BEGIN
            bit _ table + length;
            FOR table UP UNTIL >= bit DO [table] _ -1;
            RETURN;
            END;
         table _ table + bit / 36;
         R3 _ -((bit MOD 36)-1);
         R1 _ [table];
         R4 _ 400000000000B;
         !LSH 4,(3); %justify the bit for change%
         !TDO 1,4; %turn it on, thereby clearing it!%
         [table] _ R1;
         RETURN;
         END.
   %check if bit is assigned in specified table%
      (chkbit) PROCEDURE (table,bit,length);                      10E1
         table _ table + bit / 36;
         R3 _ -((bit MOD 36)-1);
         R1 _ [table];
         R4 _ 400000000000B;
         !LSH 4,(3); %justify the bit for change%
         RETURN ( NOT (R1 .A R4));
         END.
% Wheel status. %
   (enablw)PROC;                                                  11A
      LOCAL pcap;
      R1 _ 485;
      !JSYS rpcap;
      pcap _ R3;
      R3 _ R2;
      IF NOT R2 .A 685 THEN RETURN(-1);
      R1 _ 485;
      !JSYS epcap;
      RETURN(pcap)
      END.

   (disablw)PROC(pcap);                                           11B
      %Disable wheel status, and restore status to pcap%
      R1 _ 485;
      !JSYS rpcap;
      R3 _ pcap;
      R1 _ 485;
      !JSYS epcap;
      RETURN(pcap)
      END.

   (enwheel) %logged in user an enabled wheel?%
   PROCEDURE;                                                     11C
      %----------%
```

```
    LOCAL pcap;
    %----------%
    R1 _ 485;
    !JSYS rpcap;
    pcap _ R3;
    RETURN (pcap .A 485);
    END.

% programmer aids %
  (seestid)   % CL: ;  print first 72 chars of string pointed to by
  stid in R1 %
  PROCEDURE;                                                          12A
    % Declarations %
      LOCAL TEXT POINTER  tp1, tp2;
      LOCAL STRING lookstid[75];
    tp1 _ R1;
    tp1[1] _ 1;
    FIND tp1 > $72(CH) ^tp2;
    *lookstid* _ tp1 tp2;
    dismes(1,$lookstid);
    % Return %
      RETURN;
    END.

  (seeflout)   % Convert a floating point number into a string %
  PROCEDURE;                                                          12B
    LOCAL bptr;
    LOCAL STRING str[50];
    REF astr;
    R1.LH _ -1;
    R1.RH _ $str+1;
    R3 _ 0; %format word %
    IF NOT SKIP !JSYS flout THEN err($"illegal FP output");
    bptr _ R1;
    str.L _ (bptr.RH - $str)*5 - bptr.LH/70000B;
    dismes(1, $str);
    RETURN;
    END.

% Misc Routines%
  (flagut)PROCEDURE(flagno, jsysno);                                  13A
    %This routine does the equivalent of calling the old flag jsies
    using the file <netsys>nlsflags.flags;1%
    LOCAL blkadr, indx, wrd, ret;
    REF wrd;
    IF NOT blkadr _ flgpage( :indx) THEN
      CASE flagno OF
        =0: RETURN(1); %say journal is locked%
        ENDCASE RETURN(0); %say flag is false%
    &wrd _ blkadr + flagno;
    CASE jsysno OF
      =$setfg;
        BEGIN
        R1 _ 1;
        !EXCH R1,@wrd;
        ret _ IF R1 THEN 0 ELSE 1;
```

```
        END;
      =$tstfg:
        BEGIN
        ret _ wrd;
        END;
      =$rstfg:
        BEGIN
        R1 _ 0;
        !EXCH R1,@wrd;
        ret _ R1;
        END;
      ENDCASE err($"Illegal Flag Jsys");
   %map out the page%
      R1 _ -1;
      R2 _ 4B11 .V (blkadr/1000B);
      R3 _ 0;
      !JSYS pmap;
   %thaw core page%
      frzblk(indx, -1);
   RETURN(ret);
   END.

(flgpage)PROCEDURE; %map in page 0 of NLSFLAGS.FLAGS to allocated
(and frozen) page.  Open flags file if nescessary.  Return blkadr
and index (CRPGAD). RETURN 0 on failure to open%                    13B
   LOCAL blkadr, indx;
   LOCAL STRING tempsr[120];
   IF NOT flagjfn THEN
      BEGIN
      %open mailer's flag file%
         IF (flagjfn _ sgtjfn( getgtjflg(read,0,oldvrsn), $flgfname,
         $tempsr )) AND NOT (sysopen (flagjfn, rwthawed, random,
         $tempsr)) THEN reljfn(flagjfn:=0);
      END;
   IF NOT flagjfn THEN RETURN(0); %didn't get flag file open%
   %get a frozen core page%
      frzblk ((indx _ lodrfb(0,-1)), 1);
      blkadr _ crpgad[indx];
   % map in file page%
      !HRLZ R1,flagjfn;
      R2 _ 4B11 .V (blkadr/1000B);
      R3 _ 14B10;
      !JSYS pmap;
   RETURN(blkadr, indx);
   END.
(fndungtxt) %FIND unquoted text string%
PROCEDURE (keytext, start, t1, t2);                                13C
   %----------%
   LOCAL quoflg;
   LOCAL TEXT POINTER p1, p2, end;
   REF keytext, start, t1, t2;
   %----------%
   FIND start ^p2 SE(start) ^end;
   LOOP %through unquoted text segments%
      BEGIN
      FIND p2 ^p1 > ([""] ^p2 _p2 FS quoflg / end ^p2 FR quoflg);
```

```
      IF ((FIND p1 > [*keytext*] ^t2 ^t1) AND (t2 [1] <= p2 [1]))
      THEN
         BEGIN
         t1 [1] _ t1 [1] - keytext.L;
         RETURN (TRUE);
         END;
      IF quoflg THEN %flush till closing quote%
         IF NOT FIND p2 > CH ['"] ^p2 THEN EXIT LOOP;
      IF p1 [1] = end [1] THEN EXIT LOOP;
      END;
   RETURN (FALSE);
   END.

(cldtxt) PROCEDURE(ptr1, ptr2);                                    13D
   REF ptr1, ptr2;
   FIND ptr1 ^sptr1 ptr2 ^sptr2;
   <UTILTY, cldsr> ($sptr1);
   RETURN;
   END.

(bumpstr)PROC(astr);                                               13E
   %Increments number in astr by one%
   REF astr;
   %We must use a local string until L10 gets fixed%
      LOCAL STRING tempsr[10];
   *tempsr* _ STRING(VALUE(&astr)+1);
   *astr* _ *tempsr*;
   RETURN;
   END.
(makeptr)PROC(stid, ptr);                                          13F
   %Make a pointer pointing to the first character of stid, and
   store it into ptr%
   REF ptr;
   FIND SF(stid) ^ptr;
   RETURN;
   END.
(bkc) PROCEDURE (astrng);  %backspace char%                        13G
   %Given the address of an A-string, this routine will subtract one
   from the length, and then return the last character of the
   string.%
   %-------------%
   REF astrng;
   IF astrng.L <= empty+1 THEN
      BEGIN
      astrng.L _ empty;
      RETURN(ENDCHR)
      END
   ELSE BUMP DOWN astrng.L;
   RETURN(*astrng*[astrng.L]);
   END.

(bkw) PROCEDURE(astr); %backspace word%                            13H
   %BacKspace Word (VISIBLE).  The A-string, whose address is
   passed, is backed up one VISIBLE via routine bkc.%
   %-------------%
   LOCAL char;
```

```
    CASE char _ bkc(astr) OF
       =SP, =TAB, = CR, = ENDCHR : NULL;
       ENDCASE REPEAT;
    RETURN(char);
    END.

(filesc) % xxx *** new filesys %%End string construction.%
                                                                    13I
    PROCEDURE;
    LOCAL
       sdb,
       datasiz, %size of data in words%
       stdb,    %stdb for new SDB%
       initflg, % if TRUE, crepr2 uses values in dlleft and dlright
       for name delimiters %
       dlleft, %left name delimiter%
       dlright,  %right name delimiter%
       sdbit, % stid of inferior tree, if any, of existing property
       block.  It must be linked to the new block if necessary.
       (Currently, text blocks do not have inferior trees and this is
       always zero, but this code is put in just in case things
       change. %
       cl;      %pointer into clist%
    REF sdb;
    % set up byte pointer (used by apachr) to sar, the buffer string
    which was used to get the material to be copied over. %
       nsdbpt _ chbptr(sar.L) + $sar;
       IF sar.L = empty THEN apachr(SP);
       datasiz _ (sar.L+4)/5;
    % If there is currently a text block property for this node, we
    delete it after saving name delimiters and the inferior tree
    stid. %
       IF lodprop( rplsid, txttyp : &sdb, stdb) THEN
          BEGIN %get the delimiters for the statement%
          dlleft _ sdb.slnmdl;
          dlright _ sdb.srnmdl;
          initflg _ TRUE;
          sdbit _ freprop(stdb); %get rid of the old sdb%
          END
       ELSE
          BEGIN
          % let crepr2 create the name delimiters %
          dlleft _ dlright _ initflg _  sdbit _ 0;
          END;
    % create text block %
       IF NOT (stdb _ crepr2( rplsid, txttyp, datasiz, $sar+1,
       initflg, dlleft, dlright)) THEN
          err($"System error in statement edit.  Possible bad
          file.");
       % If an inferior tree existed, it must be linked into the new
       block %
          IF sdbit THEN insitree( sdbit, stdb );
    % update the correspondence list %
       UNTIL clchng = clmpty DO
          BEGIN
          POP clchng TO cl;
```

```
        [cl].clfixed _ FALSE;
        [cl].clst1 _ rplsid;
        END;
   RETURN;
   END.
```

(xtrnam) %***% %Arguments are 1: address of A-string, 2: address of work area for READC, 3: left name delimiter, 4: right name delimiter. If instead of passing the delimiters you want xtrnam to use the delimiters given in the SDB header, pass a minus one as the third argument. If finds a name in the statement it is placed in the A-string, otherwise the A-string is set to empty. Uses (TXTEDT, nmdr) to determine the name.%

13J

```
   PROCEDURE (ast, worka, dlleft, dlrght);
   LOCAL ch, sdb;
   LOCAL TEXT POINTER tp1, tp2, tp3;
   LOCAL STRING dlstr[1], drstr[1];
   REF ast, sdb;
   IF dlleft = -1 THEN %must look up the delimiters%
      BEGIN
      dlleft _ getnmdl( [worka] : dlrght);
      END;
   IF dlleft # 0 THEN *dlstr* _ dlleft;
   IF dlrght # 0 THEN *drstr* _ dlrght;
   tp1 _ [worka]; tp1[1] _ [worka+1];
   *ast* _ NULL; %clear a-string to empty%
   FIND tp1 >;
   IF dlleft = 0 OR
   (dlleft = SP AND FIND 1$SP ^tp1) OR
   FIND $SP *dlstr* $SP ^tp1 THEN
      BEGIN
      <TXTEDT, nmdr>($tp1, $tp2, $tp3);
      % if no name then tp2 = tp3 %
      FIND tp3 > ^tp1;
      IF dlrght = 0 OR
      (dlrght = SP AND FIND SP ^tp1) OR
      FIND $SP *drstr* ^tp1 THEN
         *ast* _ + tp2 tp3;
      END;
   [worka] _ tp1; [worka+1] _ tp1[1];
   fechc1(1, worka);
   RETURN
   END.
```

(txtdat) %***% PROCEDURE;

13K

```
   LOCAL space, stdb;
   IF NOT lodprop( swork, txttyp :space, stdb) THEN
      err($" No text block with this node");
   RETURN(gettim(stdb))
   END.
```

(ficpfse) %***% PROCEDURE (stid);

13L

```
   %statement end t-pointer%
   % should only be used with statements which have text property--
   error if none found %
```

```
   %called by SPL compiler only%

   LOCAL sdbloc;
   IF NOT lodprop( stid, txttyp : sdbloc) THEN
      err($"No text block associated with node");
   A2 _ [sdbloc].schars + 1;
   A1 _ stid;
   RETURN
   END.

(flfechc1) %***%                                                    13M
   %Documentation%
      %This routine is called to initialize a work area for reading
      characters from a statement.  Arguments are: direction of
      reading characters (=0 then backwards) and the address of the
      7 word work area (of which the last 2 words are no longer
      used).
      If characters are to be read from a statement then
         when calling FECHC1, the first two cells of the work area
         must contain a Tpointer.  A character count of one
         indicates the first character of the statement. FECHC1 will
         initialize the rest of the work area. The first word of the
         word area always has the PSID of the statement.  The second
         word has the character count.  The third word contains a
         bound on characters to be read.  ENDCHR'S are returned
         after reach this bound.  The fourth word has the direction
         of readout for use by readc.  A READC(x) actually results
         in the value of x being loaded into register wa followed by
         a JSP A4,readc.
         The fifth word of the work area contains a byte pointer to
         the character last read from the statement. Thus an ILDB
         instruction may be used to get the next character if the
         direction is forward.  If the direction of reading is
         backward. then the byte pointer is decremented by in-line
         code.
         To read characters from the statement execute a READC(x)
         where the value of x is the address of the work area to be
         used.  The character is returned as the value of the READC.
          Subsequent READC'S will return the following characters.
         To change position or direction within the statement the
         work area must be reinitialized by calling FECHC1 again, as
         described above.  There may however be more than one work
         area currently in use, and these may be changed
         independently.
      If characters are to be read from an A-string then
         the first word of the work area contains the address of the
         A-string instead of a PSID.  The second word is 1 if the
         first character of the string is to be read next, two if
         the second, etc.  Characters may be read out of an A-string
         in either direction, just like a statment.  Endcharacters
         are returned when the string is exhausted.%
   PROCEDURE (dir, worka);
   LOCAL
      stdb, % stdb for text block of this node %
      addr; %address of data block%
   %set work+2 to bound%
```

```
      IF NOT lodprop( [workal, txttyp : addr, stdb) THEN
         err($"No text block associated with node");
      [Worka+21 _ IF pe THEN pe ELSE [addr].schars + 1;
      addr _ addr + sdbhdl; %addr of char%
   RETURN( addr, stdb );
   END.

(idtst) %***% PROCEDURE(astrng);                                    13N
   LOCAL space, stdb;
   IF NOT lodprop( swork, txttyp, :space, stdb) THEN
      err($" No text block with this node");
   RETURN(getint(stdb) = [astrng+1])
   END.

% NUMBER CONVERSION %
   (getpint)  % convert 2 text pointers to integer %                14A
      PROCEDURE
         (tp1,  % starting text pointer %
         tp2 ); % ending text pointer %
      LOCAL STRING
         locstr[50];    % temp string %
      REF tp1, tp2;

      *locstr* _ tp1 tp2;
      FIND SF(*locstr*);
      CASE READC OF
         = D : REPEAT CASE;
         = ENDCHR : EXIT CASE;
         ENDCASE ABORT( badnum, $"Illegal Number" );
      RETURN(VALUE($locstr));
      END.

FINISH of UTILTY

% trap illegal instructions %
   (pr) PROCEDURE % print register values for illegal instruction trap
   pseudo-interrupts %                                              18A
      (astr,  % address of leading string to print %
      val     % value associated with the string %
      );
      REF astr;
      % type leading string %
         typeas( $"
         ");
         typeas( &astr);
      % type value symbollically %
         typval( val, 0);
      % now type it in octal %
         typeas( $" =");
         typval( val, 1);
      % now return %
         RETURN;
      END.
% Don't want stack overflow or error handling routines anymore - so
comment them out ----
%% stack overflow handling %%
```

```
EXTERNAL stkovr, sysovr;
(xyzqwe) PROCEDURE; %% dummy procedure %%                              20B
   %% stack overflow from start of procedure %%
      (sysovr): NULL;                                                  20B1A
   %% stack pointer overflow interrupt code%%
      (stkovr): %%process stack overflow%%                             20B2A
         %%initialize stack pointers%%
            %%general call stack%%
               S _ -$gstksz; !HRL S,S; !HRRI S,gstack;
               M _ S;
               gstack _ $uflow;
            %%pattern stack%%
               p _ -$pstksz; !HRL p,p; !HRRI p,pstack;
            %% spec stack %%
               spsk _ spsk1;
            %% BACKEND %%%% sequence generator stacks %%
               sqinit();
         !MOVE A1,@127B; !LSH A1,-23;
         CASE A1 .A 4M OF
            =$S;
               werr(2); %%print S-stack overflow and wait for
               CA%%
            =$p;
               werr(6); %%print p-stack overflow and wait for
               CA%%
            ENDCASE
               werr(3);
                  %%print unknown stack overflow and wait for
                  CA%%
   END.

%%error handling routines %%
   (deferr)PROC;                                                       21A
      %%This procedure gets called when a signaal is done with nothing
      stoping it on the stack.
         If the current stack is not the system stack, GSTACK, then a
         search is made of sequence generator work areas for one which
         liinks this stack to another.  If one is found, M and S are
         loaded from it and the signal is propagated through that
         stack.  If no link can be established, an error message is
         typed and we let it happen.
         If the current stack is the system stack, it types out the
         message in sysmsg, and resets NLS%%
      IF (sysmsg <= errmsbase)  THEN *sar*  _ "Error"
      ELSE *sar* _ *[sysmsg]*; %%Move message to a safe place%%
      sysmsg _ $sar;
      IF M.RH NOT IN [$gstack, $gstack+$gstksz] THEN
         BEGIN %%must propagate signal further%%
         %%check seqgen stacks%%
         R4 _ $sqstks;
         FOR R5 _ $sqgwas UP $sqwrkl UNTIL >= $sqgaend DO
            IF [R5].swalloc AND M.RH IN [R4, R4+$sqstksz] THEN
               BEGIN
               M _ [R5].swmrsav;
               S _ [R5].swsrsav;
               GOTO syssig;
```

```
            END
         ELSE R4 _ R4+$sgstksz;
      %%could not find link in seggen work areas, so let it rip%%
         dismes(2, $"DEFERR called with other than GSTACK--no
         linkage found to GSTACK (please report this to ARC
         programming personell)");
      END;
   IF nlmode # typewriter THEN <INPFBK, bmoff>();
   CASE sysgnl OF
      = statesig: supervisor();
      =0, =-2: %%called by err, abort%%
         dismes(2, $sar);
      =-1, =-3: %%called by werr, wabort%%
         BEGIN
         *sar* _ *sar*, "--Type CA";
         clrbuf(1); %% clear input and output buffers %%
         dismes(1, $sar);
         LOOP IF input() = CA THEN EXIT;
         dismes(0);
         END;
      =-5: %%Called by badfil%%
         BEGIN
         delfil(bfilno);
         IF nlmode = fulldisplay THEN
            alldsp()  %%make sure user sees real state of the
            world%%
         ELSE typeas($"File closed: ");
         REPEAT(-1);
         END;
      =-6: %%Called by iodaterr%%
         REPEAT(sysgnl _ -1); %%make it a werr%%
      ENDCASE dismes(2, $sar);
   sysmsg _ $"NLS Internal System Error";
   sysgnl _ 0;
   supervisor();
   halt();
   END.
END OF COMMENTED OUT CODE  %
```

```
< NINE, VERIFY.NLS.3, >, 6-Jul-77 13:24 KJM ;;;;<NLS>VERIFY.NLS;2,
30-MAR-72 13:39 HGL ;
FILE verify % <ARCSUBSYS>XL10  <RELNINE>verify % %  (arcsubsys,xl10,)
(RELNINE,verify.rel,) %

%control%

   (vrmain) %***% PROCEDURE (fileno, vrfy);                              4A
      % values of vrfy %
         % 0 -- Checksum %
         % 1 -- Verify %
      LOCAL
         statsr, %keep each result just for fun%
         statsd,
         ringsf, %total rings found in structure check%
         sdbsf,  %total sdbs found in structure check%
         stid;   %work pointer for file origin%
      IF vrfy THEN dismes (1, $"File Verify in Progress");
      statsr _ crng(vrfy, fileno); %total rings found in the structure
      pages%
      statsd _ csdb(vrfy, fileno); %total sdbs found in the data pages%
      stid _ 0;
      stid.stfile _ fileno;
      stid.stpsid _ origin;
      ringst _ ckstrc(stid: sdbsf);
      IF statsr # ringsf OR statsd # sdbsf THEN badfil(fileno);
      IF vrfy THEN dismes(0);
      RETURN;
      END.

% procedures for checking data blocks %

   (crng) %***% PROCEDURE (vrfy, fileno);  % ring blocks %              6A
      LOCAL
         stats,   %total number of ring elements in file%
         statb,   %total number of ring elements in block%
         rngblk,  %index in RNGST%
         rn,      %pointer to RNGST entries%
         stid,    %STID for getting blocks loaded by lodrng%
         blkad,   %address of ring block in core%
         nringl,  %number of ring elements in a block%
         frecnt,  %number of elements on the free list%
         freep,   %pointer to free list%
         blkusd;  %used word count for the block%
      REF rn;
      &rn _ filhdr(fileno) + $rngst - $filhed;
      nringl _ (blksiz-fbhdl)/ringl;
      rngblk _ stid _ stats _ 0;
      stid.stfile _ fileno;
      DO BEGIN
         IF rn # 0 THEN
            BEGIN
            stid.stblk _ rngblk;
            lodent( stid, rngtyp : blkad);
            IF vrfy THEN
               BEGIN
```

```
            %check free list%
            frecnt _ 0; %number of entries on the free list%
            IF (freep _ rn.rffree) # 0 THEN
               BEGIN
               IF freep NOT IN [fbhdl,blksiz) THEN
                  badfil(fileno);
               freep _ freep + blkad;
                  %address of start of free list%
               LOOP
                  BEGIN
                  BUMP frecnt; %count entries on free list%
                  %check if have reached end of list%
                  CASE [freep] OF
                     =0: EXIT;
                     NOT IN [fbhdl,blksiz): badfil(fileno);
                     ENDCASE IF frecnt > nringl THEN
                        badfil(fileno);
                  freep _ blkad + [freep];
                  END
               END;
            %check used word count%
            statb _ nringl - frecnt;
            stats _ stats + statb;
            blkusd _ statb * ringl + fbhdl;
            IF blkusd # rn.rfused THEN badfil(fileno);
            END
         END;
      BUMP &rn;
      END
   UNTIL (rngblk _ rngblk+1) = rngm;
   RETURN (stats) END.

(csdb) %***% PROCEDURE(vrfy, fileno); % statement data blocks %     6B
   LOCAL
      stats,  %number of nongarbage SDB's in file%
      sdbblk, %index in DTBST%
      dt,     %pointer to DTBST entry%
      stdb,   %STDB for getting blocks loaded by lodsdb%
      blkad,  %address of the block%
      freep,  %address of free space%
      sdbpt,  %pointer to SDB's in the block%
      sdbusd; %used word count in the block%
   REF dt;
   &dt _ filhdr(fileno) + $dtbst - $filhed;
   sdbblk _ stdb _ stats _ 0;
   stdb.stfile _ fileno;
   DO BEGIN
      IF dt # 0 THEN
         BEGIN
         stdb.stblk _ sdbblk;
         lodent( stdb, sdbtyo: blkad);
         IF vrfy THEN
            BEGIN
            %check free space pointer%
            IF (freep _ dt.rffree)
               NOT IN [fbhdl,blksiz] THEN badfil(fileno);
```

```
            freep _ freep+blkad; %address of free space%
            sdbpt _ blkad+fbhdl; %address of first SDB%
            sdbusd _ fbhdl; %used word count%
            WHILE freep > sdbpt DO
               BEGIN
               IF NOT [sdbpt].sgarb THEN
                  BEGIN
                  sdbusd _ sdbusd + [sdbpt].slength;
                  BUMP stats;
                  END;
               %address of next SDB%
               IF (sdbpt := sdbpt + [sdbpt].slength) >= sdbpt THEN
               badfil(fileno);
               END;
            IF sdbusd # dt.rfused THEN badfil(fileno);
            END;
         END;
      BUMP &dt;
      END
   UNTIL (sdbblk _ sdbblk+1) = dtbm;
   RETURN (stats) END.

% procedure for checking structure %
  (ckstrc) %***% PROCEDURE(orgstd);                                    7A
      %  ORGSTD is the stid of the origin of the file  or of an
      inferior tree %
      LOCAL
         fileno, %file number%
         statd,  %number of data blocks in structure%
         statr,  %number of rings in structure%
         sdbsf,
         curpty,
         cur,    %STID of current statement%
         cursup, %STID of up of current statement%
         clhead, %head flag value expected for cur%
         nxtcur, %STID of next statement%
         sdb,    %location of SDB%
         curlev, %level in structure of cur%
         toplevf, %flag called with file origin%
         looktxtf, %flag looking for text block on this node%
         sdbtabn, %sdb type order number (from getptab)%
         locstk[67]; % used to save away old value of fvstk if
         necessary %
      REF sdb;
      curlev _ statr _ statd _ 0;
      cursup _ cur _ orgstd;
      fileno _ orgstd.stfile;
      toplevf _ IF orgstd.stpsid = origin THEN TRUE ELSE FALSE;
      RESET fvstk;
      PUSH orgstd ON fvstk;
      clhead _ TRUE;
      LOOP %once thru for each ring%
         BEGIN
         BUMP statr;
         looktxtf _ toplevf; %look for text block on top level%
         %Check the property sdbs of each ring if it is not the origin
```

```
        of an inferior tree%
        IF NOT getorf(cur) THEN
           BEGIN
           curpty _ getsdb(cur); % get first property %
           sdbtabn _ -1;
           LOOP
              BEGIN
              IF curpty.stpsdb = 0 THEN EXIT;
              BUMP statd;
              lodent(curpty, sdbtyp: &sdb);
              %Hooked in the ring ok?%
              IF sdb.sgarb OR sdb.spsid # cur.stpsid THEN
              badfil(fileno);
              curpty.stpsdb _ sdb.spsdb;
              IF (sdbtabn := getptab(sdb.sptype)) >= sdbtabn THEN
              badfil(fileno);
                 %sdb's must be ordered, at most one of each type%
              %Additional property type checks follow%
              CASE sdb.sptype OF
                 =txttyp: looktxtf _ FALSE;
                 =gtftyp, =lwtyp: NULL;
                 =dhtyp, =chtyp, >=40000B:
                    IF sdb.sitpsid THEN BEGIN % must verify inferior
                    tree %
                    % Save away current verify stack; it will be
                    replaced by a new one when ckdgm calls ckstrc %
                    mvbfbf($fvstk, $locstk, svmxlev + 3 % max levels
                    plus stack overhead %);
                    statr _ statr + ckdgm(&sdb, fileno: sdbsf);
                                                          7A11E4J3D
                    % restore saved verify stack %
                    mvbfbf($locstk, $fvstk, svmxlev + 3 % max levels
                    plus stack overhead %);
                    statd _ statd + sdbsf;
                    END;
                 ENDCASE badfil(fileno); % Illegal property type %
              END;
           END;
        IF looktxtf THEN badfil(fileno);
           %should have found text, didn't%
        %check head flag%
        IF clhead # getfhd(cur) THEN badfil(fileno);
        IF (nxtcur _ getsub(cur)) # cur THEN
           BEGIN %go down in structure%
           IF (curlev _ curlev+1) > svmxlev  THEN badfil(fileno);
           PUSH cursup ON fvstk;
           cursup _ cur;
           cur _ nxtcur;
           clhead _ TRUE;
           END
        ELSE %go to successor%
           BEGIN
           nxtcur _ getsuc(cur);
           WHILE nxtcur = cursup AND nxtcur # orgstd DO
              BEGIN
              %must have tail flag set%
```

```
            IF getftl(cur) = FALSE THEN badfil(fileno);
            cur _ nxtcur;
            nxtcur _ getsuc(cur);
            POP fvstk TO cursup;
            curlev _ curlev-1;
            END;
         cur _ nxtcur;
         %head flag must be off%
         clhead _ FALSE;
         END;
      IF cur = orgstd THEN RETURN (statr, statd);
      END;
   END.

% specific procedures for checking properties and their substructure%

   (ckdgm) %***% PROCEDURE (sdbloc, fileno); %check diagrams and
   subtrees%
      LOCAL
         statr,   %total number of rings%
         statd,   %total sdbs%
         stid;    %work pointer to subtree heads%
      %check the inferior tree%
      IF NOT (stid _ [sdbloc].sitpsid) THEN                          9A3
         RETURN( 0, 0);
      stid.stfile _ fileno;                                          9A4
      statr _ ckstrc(stid:statd);
      RETURN(statr, statd) END.

FINISH of verify
```

(dontset)	<nine, wmdata, 016>	EXT CONSTANT =0	2B
(doset)	<nine, wmdata, 014>	EXT CONSTANT =1	2A
(rdebug)	<nine, wmdata, 02>	EXT	3
(remailbox)	<nine, wmdata, 05>	EXT STRING	8
(nswnode)	<nine, wmdata, 012>	EXT STRING	6
(nswpplusn)	<nine, wmdata, 011>	EXT STRING	7
(nswproject)	<nine, wmdata, 04>	EXT STRING	5
(nswuser)	<nine, wmdata, 03>	EXT	4
(wmmailbox)	<nine, wmdata, 06>	EXT STRING	9

```
< NINE, WMDATA.NLS;3, >, 15-May-78 13:31 SKO ;;;;
FILE wmdata  % <arcsubsys,x110,> TO <relnine,wmdata.rel,>%
%qset values for WM calls%
   (doset) EXTERNAL CONSTANT = 1;                                          2A
   (dontset) EXTERNAL CONSTANT = 0;                                        2B
(idebug) EXTERNAL = 0;  %debug switch for WM file interface%                3
(nswuser) EXTERNAL ;  %nsw user-id%                                         4
(nswproject) EXTERNAL STRING [12];  %nsw project name%                      5
(nswnode) EXTERNAL STRING [12];  %nsw node name%                            6
(nswppiusn) EXTERNAL STRING [25];  %nsw project+node name%                  7
(femailbox) EXTERNAL STRING _ "FE   ";                                      8
(wmmailbox) EXTERNAL STRING _ "WM   ";                                      9
FINISH wmdata                                                              10
```

```
< NINE, NLS-GRAMMAR.NLS;77, >, 3-Jun-78 12:34 KIRK ;;;;; EXTERNAL
LINKS: <nine,sysgd,>
% This file contains the grammar constructs (declarations, rules and
commands) used by NLS 9 grammars.  It is not a compilable grammar file
but contains links referenced in INCLUDE statements in NLS 9 subsystem
grammar source files (including BASE) %
% FLAGS %                                                          2
   %Shared page%                                                   2A
      SET DPS=FALSE;  %Set to TRUE for DPS protocol%
      SET SHARED=TRUE;  %Set to TRUE if shared page protocol%
      SET NSW= FALSE;  %Set to TRUE  if in NSW%
   %NSW%                                                           2B
      SET DPS=FALSE;  %Set to TRUE for DPS protocol%
      SET SHARED=FALSE;  %Set to TRUE if shared page protocol%
      SET NSW= TRUE;  %Set to TRUE  if in NSW%
   %Base%                                                          2C
      SET UOWORD=FALSE;  % Do not compile useroption command words %
      SET BASESUB=TRUE;  %Set to FALSE for subystems%
   %Subsystems%                                                    2D
      SET BASESUB=FALSE;  %Set to FALSE for subystems%
% DECLARATIONS %                                                   3
   % UNIVERSAL (useable in all subsystems) %                      3A
      % GLOBALS % DECLARE GLOBAL                                  3A1
         entsubsystem, %entry subsystem (string variable)%
         insmode,  %TRUE if User is in insert mode%
         ttysim,  %TRUE if this TTY is simulated%
         subslist, readsubs;  % Goto Subsys support %
      % PARSEFUNCTIONS % DECLARE PARSEFUNCTION                    3A2
         editstring, levadj, viewspecs, cmntpf, msbtvs, feprocess,
         fenewsubsys, feendsubsys, feexcm, feterminate, felogout,
         pfsubc, pfsubname, pfsubs, pfinss, pfjmpi, pfjmps, pfclrwnd,
         lookcd, hlpset, hlplook, setty, rstty, simtty, simdisplay,
         fedelsubsys , feusrid, mustconfirm;
      DECLARE COMMAND WORD
         % selectors (others under Base) %                        3A3A
            "NEWFILENAME"=6 SELECTOR
               POINT = pbranch TYPEIN = adr1fo ADDRESS = adr1fo,  3A3A1A
            "OLDFILENAME" = 7 SELECTOR
               POINT = pbranch TYPEIN = adr1fo ADDRESS= adr1fo,   3A3A2A
            "FILENAME" = 12 SELECTOR
               POINT = pbranch TYPEIN = adr1fo ADDRESS= adr1fo,   3A3A3A
            "CHARACTER" = 2 SELECTOR
               POINT = CHARACTER TYPEIN = CHARACTER ADDRESS= adr1fo,
                                                                  3A3A4A
            "WORD" = 3 SELECTOR
               POINT = WORD TYPEIN = WORD ADDRESS= adr1fo,        3A3A5A
            "TEXT" = 1 SELECTOR
               POINT = TEXT TYPEIN = TEXT ADDRESS= adr2fo,        3A3A6A
            "ADDRESS" = 1 SELECTOR
               TYPEIN = adr1fo,                                   3A3A7A
            "VISIBLE" = 4 SELECTOR
               POINT = VISIBLE TYPEIN = TEXT ADDRESS= adr1fo,     3A3A8A
            "INVISIBLE"= 11 SELECTOR
               POINT = INVISIBLE TYPEIN = TEXT ADDRESS= adr1fo,   3A3A9A
            "LINK" = 30 SELECTOR %should be FILENAME%
               POINT = pbranch TYPEIN = adr1fo ADDRESS= adr1fo,   3A3A10A
```

```
      "NUMBER" = 8 SELECTOR
         POINT = pbranch TYPEIN = tnum ADDRESS= adr1fo,     3A3A11A
      %note, "number" is handled in the back-end.  it is the same
      concept as "number" in NLS 8.5..  this should change when
      routines fixed%
      "INTEGER" SELECTOR = INTEGER,
      "PASSWORD" = 10 SELECTOR = PASSWORD,
      "STATEMENT" = 29 SELECTOR
         POINT = pbranch TYPEIN = TEXT ADDRESS= adr1fo,     3A3A15A
      "LOCATION" = 29 SELECTOR
         POINT = pbranch,                                   3A3A16A
      "BRANCH" = 26 SELECTOR
         POINT = pbranch TYPEIN = TEXT ADDRESS= adr1fo,     3A3A17A
      "PLEX" = 28 SELECTOR
         POINT = pbranch TYPEIN = TEXT ADDRESS= adr1fo,     3A3A18A
      "GROUP" =27 SELECTOR
         POINT = pgroup TYPEIN = TEXT ADDRESS = adr2fo,     3A3A19A
      "NAME" = 32 SELECTOR
         POINT = pbranch TYPEIN = TEXT ADDRESS= adr1fo,     3A3A20A
      "WINDOW" = 33 SELECTOR
         POINT = pgetcoords,                                3A3A21A
      "ITEM" = 26 SELECTOR
         POINT = CHARACTER,                                 3A3A22A
      "IDENT" = 121 SELECTOR
         POINT = pbranch TYPEIN = TEXT ADDRESS= adr1fo,     3A3A23A
    % commandwords %                                          3A3B
      "BACK" = 59,
      "CONTENT" = 53,
      "CNTLQ" = 51,
      "DOWN" = 55,
      "END" = 58,
      "EXTERNAL" = 59,
      "EXTNAME" = 66,
      "FILE" = 51,
      "FILENAMED" = 63,
      "FILERETURN" = 67,
      "FIRST" = 53,
      "FIRSTCONTENT" = 68,
      "FIRSTNAME" = 64,
      "FIRSTWORD" = 70,
      "GRAMMAR" = 55,
      "HEAD" = 56,
      "IDENTLIST" =13 SELECTOR = TEXT,
      "NEXT" = 61,
      "NEXTCONTENT" = 69,
      "NEXTNAME" = 65,
      "NEXTWORD" = 71,
      "OFF" = 63,
      "ON" = 62,
      "ORIGIN" = 60,
      "PREDECESSOR" = 53,
      "RETURN" = 62,
      "SUCCESSOR" = 52,
      "TAIL" = 57,
      "UP" = 54,
      "VIEWSPECS" = 57;
```

```
   % FUNCTIONS: XROUTINES %
      DECLARE FUNCTION  xgetjumpring, xjmpcnt, xjumpreturn,
      xjumpaddr, xjump, xhandle, xgoto, xmousespecs, xsubsget,
      xquitsubsys, xquit, %HELP RULE SUPPORT% helpinit, helpshow,
      checkmore, rstmore, helpring, helpterm;
   % VARIABLES % DECLARE VARIABLE                                    3A5
      lastcontent,  %last content for search commands%
      ent, type, source, dest, level, vs, sent, dent, addr, param,
      param2, param3, param4, keepgoing, index, rtnlist, sqkvar,
      alltext;
% BASE %                                                             3B
   DECLARE COMMAND WORD
      % Selectors %                                                  3B1A
         "DIRECTORY" = 9 SELECTOR
            POINT = pbranch  TYPEIN = adr1fo  ADDRESS = adr1fo,
      % commandwords %                                              3B1B
         "ACCESSES" = 87 ,
         "ACCOUNT" = 73 ,
         "ALL" = 52,
         "ALLOW" = 89,
         "ALPHABETICAL" = 79 ,
         "APPEND" = 84 ,
         "ARCHIVE" = 61,
         "ARCHIVED" = 94 ,
         "ASSEMBLER" = 56,
         "BOTH" = 68 ,
         "BOTTOM" = 53,
         "BUFFER" = 58,
         "BYTESIZE" = 93 ,
         "CASE" = 52,
         "CENTER" = 54,
         "CHARATATIME" = 55,
         "COM" = 55,
         "COMTSTTY" = 59,
         "COMP80" = 50,
         "COMPACT" = 53,
         "CREATION" = 78 ,
         "CURCONTEXT" = 51,
         "DATE" = 59,
         "DEFAULT" = 54,
         "DEFERRED" = 52,
         "DELETE" = 50,
         "DISK" = 52,
         "DISPLAY" = 51,
         "DUMP" = 83 ,
         "EDGE" = 55,
         "EVERYTHING" = 84 ,
         "EXECUTE" = 83 ,
         "EXECUTIVE" = 55 ,
         "EXTENSION" = 76 ,
         "FOR" = 70 ,
         "FORBID" = 71 ,
         "FORMATTED" = 54,
         "FROZEN" = 51,
         "HALFDUPLEX" = 53,
         "HLPCOM" = 52,
```

```
"HORIZONTALLY" = 51,
"INDENTING" = 55,
"INPUT" = 51,
"JOURNAL" = 53,
"JUSTIFIED" = 54,
"L10" = 52,
"LAST" = 51,
"LEFT" = 52,
"LENGTH" = 85 ,
"LEVELADJUST" = 53,
"LINEATATIME" = 54,
"LIST" = 85 ,
"LOWER" = 52,
"MARKER" = 51,
"MISCELLANEOUS" = 86 ,
"MODE" = 51,
"MODIFICATIONS" = 53,
"NEW" = 51,
"NO" = 74 ,
"NOT" = 53,
"OLD" = 52,
"ONE" = 51,
"OPTION" = 59,
"OUTPUT" = 52,
"PAGE" = 54,
"PREVENT" = 54,
"PRINTER" = 54,
"PRINTOPTIONS" = 55,
"PRIVATE" = 51,
"PROCEDURE" = 54,
"PROGRAM" = 52,
"PROGRAMMER'S" = 60,
"PROGRAMS" = 52,
"PROTECT" = 88 ,
"PUBLIC" = 54,
"QUICKPRINT" = 51,
"READ" = 81 ,
"REAL" =  9,
"REMOTE" = 58,
"RENAME" = 54,
"RESET" = 55,
"REST" = 52,
"REVERSE" = 77 ,
"RIGHT" = 51,
"SELF" = 70 ,
"SENDMAIL" = 53,
"SENTENCE" = 54,
"SEQGEN" = 51,
"SEQUENTIAL" = 52,
"SET" = 92 ,
"SINGER" = 51,
"SIZE" = 55,
"SORT" = 57,
"STARTUP" = 56,
"STATUS" = 52,
"TAB" = 56,
```

```
      "TAPE" = 72 ,
      "TEMPORARY" = 56,
      "TENEX" = 55,
      "TERMINAL" = 57,
      "TIME" = 58,
      "TTY" = 52,
      "TWO" = 53,
      "TYPEWRITER" = 52,
      "UNDELETE" = 69 ,
      "UNFORMATTED" = 55,
      "UPPER" = 51,
      "USER" = 59,
      "VERBOSE" = 91 ,
      "VERSIONS" = 75 ,
      "VERTICALLY" = 52,
      "800VIDEOCOMP" = 52,
      "500VIDEOCOMP" = 53,
      "WRITE" = 82 ;
   %+PROGRAMMERS%                                                    3B1C
   DECLARE COMMAND WORD                                              3B1D
      "CATCHPHRASE" = 102,
      "COMMENT" = 103,
      "COROUTINE" = 104,
      "DO-UNTIL" = 105,
      "DO-WHILE" = 106,
      "GRAMMAR" = 108,
      "IF-THEN-ELSE" = 109,
      "LOOP" = 110,
      "PARSEFUNCTION" = 111,
      "UNTIL-DO" = 113,
      "WHILE-DO" = 114,
      "SUBSYSTEM" = 115,
      "SEQUENCE-GENERATOR" = 116;
   %+PROGRAMMERS%                                                    3B1E
   %+UOWORD%  %These command words must be given values < = 127%
                                                                     3B1F
   DECLARE COMMAND WORD                                              3B1G
      %for USEROPTION. Those which interact with NLS are
      commented out until seperation%
      "TYPEINT" = 100 SELECTOR TYPEIN = INTEGER,
      "TYPETXT" = 101 SELECTOR TYPEIN = TEXT ,
      "ANTICIPATORY" = 121,
      "BC" = 122,
      %"BOTTOM" = 123,%
      "BS" = 124,
      "BW" = 125,
      "CA" = 126,
      "CD" = 127,
      %"CHARACTER" = 128,%
      "DEMAND" = 129,
      "UENTRY" = 130,
      "UXCLUDE" = 165,
      "EXECUPORT" = 131,
      "UFILERETURN" = 132,% comment out when U is removed %
      "FIXED" = 133,
      "FULL" = 134,
```

```
            "IGNORE" = 135,
            "IMLAC" = 136,
            "UINCLUDE" = 137,
            %"INDENTING" = 138,%
            "INSERT" = 139,
            %"LEFT" = 140,%
            "LINEPROCESSOR" = 142,
            %"LINK" = 143,%
            "LITESC" = 144,
            %"MODE" = 145,%
            %"NUMBER" = 146,%
            %"OFF" = 147,%
            %"OLDFILELINK" = 148,%
            %"PAGE" = 149,%
            "PARTIAL" = 150,
            %"PROGRAM" = 151,%
            "URETURN" = 152,% comment out when U is removed %
            %"RIGHT" = 153,%
            "RPT" = 154,
            "SC" = 155,
            "SUBSYSTEM" = 156,
            "SW" = 157,
            %"TAB" = 158,%
            "TASKER" = 159,
            "TERSE" = 160,
            %"TEXT" = 161,%
            "TI" = 162,
            "VERBOSE" = 163,
            "WRAPAROUND" = 164;
         %+UOWORD%                                                    3B1B
      DECLARE FUNCTION PROCESS = "CRT NLSBE", PACKAGE = "EDITOR":
         %BASE XROUTINES% xupdate, xverify, xtranspose, xsubstitute,
         xsubresolve, xsort, xsimulate, xshow, xset, xrun, xreset,
         xreplace, xrepeatsearch, xrenumber, xrelease, xputtext,
         xprtstmt, xprtcc, xprtsnum, xprtprev, xprtnext, xprint,
         xpcompile, xoutput, xopen, xnswinit, xmove, xmerge, xmark,
         xload, xinsert, xinsstatement, xgettext, xfreeze, xforce,
         xfterm, xexpand, xestablish, xdisestablish, xdelete, xcreate,
         xcopy, xclose, xclear, xchksimtty, xbreak, xappend, xinit,
         xconnect, xlogout, xtrim, xundelete, xexpunge, xprocess,    3B2A
         %+PROGRAMMERS%                                              3B2B
         xpinsert,
         %+PROGRAMMERS%                                              3B2D
         %USEROPTIONS XROUTINES% xuocontchar, xuocurcon, xuodsply,
         xuoinclude, xuoextn, xuofeedback, xuoherald, xuonamedel,
         xuooutput, xuoprint, xuoprompt, xuorecognition, xuoreset,
         xuoringsize, xuoshow, xuostup, xuovuspc;
      DECLARE VARIABLE
         shwstr, namfil, dtype, port, ff, tfil, sim, pb, opttysim,
         termsim;
% RULES COMMONLY USED %                                                4
   % UNIVERSAL RULES AVAILABLE IN ALL SUBSYSTEMS %                     4A
      % ENTITY DEFINITIONS %
         editentity = textent / structure;                           4A1A
      % TEXT ENTITY DEFINITIONS %
         textent = text1 / "TEXT" / "LINK" / "NUMBER";               4A2A
```

```
      text1 = "CHARACTER" / "WORD" / "VISIBLE" / "INVISIBLE";          4A2B
   % STRUCTURE ENTITY DEFINITIONS %
      structure = "STATEMENT" / notstatement;                          4A3A
      notstatement = "GROUP" / "BRANCH" / "PLEX" ;                     4A3B
   % SHOW RESULT %
      showresult =                                                     4A4A
         ( IF RESULT
            ( IF DISPLAY SHOWCONFIRM(RESULT)
            / IF NOT DISPLAY SHOW(RESULT) )
         / IF NOT RESULT );
   % SWITCH %
      switch = ("ON"! 1!/"OFF"!2 L2!);                                 4A5A
   % TERMINATORS %
      seterm = TERMINATORS _ #"<^E>";                                  4A6A
   % IDENT SUPPORT HELPRULE %                                          4A7
      shwhelprule =  % help for selecting ident from username %        4A7A
         (IF HELPCODE = 101 param _ ANSWER
         / IF HELPCODE = 102 <"Is this the correct one?"> param _
         ANSWER
         / IF HELPCODE = 103 <"Type the correct IDENT:">
            param _ LSEL(#"TEXT")
         )
         RESUME(param);
% PROGRAMMER ENTITIES %                                               4B
   temptype = "CASE" !L2! / "CATCHPHRASE" !L2! /                       4B1
      "COROUTINE" !L2! / "DO-UNTIL" / "DO-WHILE" !L2! / "FOR" /
      "GRAMMAR" / "IF-THEN-ELSE" /
      "LOOP" / "PARSEFUNCTION" !L2! / "PROCEDURE" /
      "SUBSYSTEM" / "SEQUENCE-GENERATOR" !L2! /
      "UNTIL-DO" / "WHILE-DO" ;
% COMMANDS %                                                           5
   % UNIVERSAL COMMANDS AVAILABLE IN ALL SUBSYSTEMS %                  5A
      % SQUEAK rule %                                                 5A1
      SQUEAK sq =  sqkvar _ msbtvs()                                   5A1A
         xmousespecs( sqkvar, WINDOW);

      qhelp =  % help button %                                         5A2
         CLEAR <"Searching HELP file">                                 5A2A
            hlpset()                                                  5A2A1
            helpinit(pfsubname(),  %current subsystem name%           5A2A2
               #"NBASE", % the context: should be HELPSTRING. %
               NULL)                                                  5A2A2B
            helploop;

      help COMMAND =             "HELP"                                5A3
         helpbody;

         helpbody =              % body of help rule %                 5A3B
            hlpset()                                                  5A3B1
            <"Type a term and hit OK, or just hit OK">
            (CONFIRM
               param _ NULL % no user typein %                        5A3B3A
            /  param _ LSEL(#"LINK") )
            CLEAR <"Searching HELP file">                             5A3B5
            helpinit(pfsubname(),  %current subsystem name%           5A3B6
               #"BASE", % the context: should be HELPSTRING. %
```

```
            param)  % The user type-in or NULL %                    5A3B6B
         helploop;

      helploop =                                                     5A3C
         ( IF DISPLAY hchk
         / IF NOT DISPLAY
         % Get next "command" %
         CLEAR                                                       5A3C4
         helpdisp ) helploop ;

      hchk =                                                         5A3D
         param _ checkmore() % is there more to the menu? %          5A3D1
         ( IF param
            CLEAR <"do you want to see the rest of the menu?">
            ANSWER                                                   5A3D2A
               helpshow(#"NEXT", NULL, NULL) [hchk]                  5A3D2A1
         /  rstmore() );

      helpdisp  =                                                    5A3E
         keepgoing _ TRUE                                            5A3E1
            ( ("<" / "_") <"go back">
               param _ #"BACK"                                       5A3E1A1
               param2 _ 999 % initial value %                        5A3E1A2
               loopback % PERFORM bckrng UNTIL (ANSWER)%
            / "^" <"go up">
               param _ #"UP"                                         5A3E1B1
               param2 _ NULL                                         5A3E1B2
               param3 _ NULL                                         5A3E1B3
            / param _ #"NAME"
               param2 _ LSEL(#"LINK")                                5A3E1C1
               param3 _ NULL                                         5A3E1C2
            )
            [IF keepgoing helpshow(param, param2, param3)];

         loopback =                                                  5A3F
            helpring(param2 -> keepgoing, param2, param3)            5A3F1
            [IF keepgoing (ANSWER / loopback)];

   helpnd COMMAND = % terminate HELP %                               5A4
      hlplook() helpterm();
                                                                     5A4A
   goto COMMAND = "GOTO"                                             5A5
      gotobody;

      gotobody =                                                     5A5B
         <"subsystem"> subsrule
            CONFIRM                                                  5A5B1A
            xgoto( ent, sent -> dent, rtnlist)                       5A5B1B
            % newsubsrule %
            param _ fenewsubsys(dent)                                5A5B1D
            IF NOT param xquitsubsys() % tell BE if load failed % ;
                                                                     5A5B1E
         subsrule =                                                   5A5B2
            subslist _ xsubsget()  % subsrule %                      5A5B2A
            sent _ (ent _ TRUE CW: subslist /                        5A5B2B
               OPTION ent _ FALSE                                    5A5B2B1
```

```
                <"subsystem name">
                LSEL(#"OLDFILENAME"));
                                                                    5A5B2B1B
      % subsrule =
         (IF readsubs
            readsubs _ FALSE                                        5A5B3A1
            subslist _ xsubsget())                                  5A5B3A2
         / (IF NOT readsubs TRUE);

      newsubsrule =                                                 5A5B4
         (IF rtnlist  subslist _ rtnlist)
         / (IF NOT rtnlist TRUE);
         %
execute COMMAND = "EXECUTE" <"command in">                          5A6
   executebody;

   executebody =                                                    5A6B
      subsrule
      xgoto( ent, sent -> dent, rtnlist)                            5A6B2
      feexcm(dent)                                                  5A6B3
      xquitsubsys();
                                                                    5A6B4
ddt COMMAND =                                                       5A7
   "DDT" !L2! <"type <ESC>p to exit ddt"> CONFIRM
   xhandle(WINDOW);

jump COMMAND = "JUMP" <"to">                                        5A8
   jumpbody;

   jumpbody =                                                       5A8B
      (IF DISPLAY
         ( NULL
            ent _ #"STATEMENT"                                      5A8B1A1
            dest _ DSEL( #"LOCATION" )                              5A8B1A2
            vs _ viewspecs()                                        5A8B1A3
            CONFIRM                                                 5A8B1A4
            xjump( ent, dest, vs, WINDOW )                          5A8B1A5
         / "ADDRESS" <"relative to">
            ent _ #"STATEMENT"                                      5A8B1B1
            dest _ DSEL( #"STATEMENT" )                             5A8B1B2
            <"modified by address">
            addr _ LSEL( #"ADDRESS" )                               5A8B1B4
            vs _ viewspecs()                                        5A8B1B5
            CONFIRM                                                 5A8B1B6
            xjumpaddr( #"STATEMENT", dest, addr, vs, WINDOW )
                                                                    5A8B1B7
         )
      / IF NOT DISPLAY
         "ADDRESS"
            ent _ #"STATEMENT"                                      5A8B2A1
            dest _ DSEL( #"CHARACTER" )                             5A8B2A2
            vs _ NULL                                               5A8B2A3
            CONFIRM                                                 5A8B2A4
            xjump(ent, dest, vs, WINDOW)                            5A8B2A5
      / jmpcoms
         CONFIRM                                                    5A8B3A
```

```
      xjump( ent, dest, vs, WINDOW )                          5A8B3B
   / jmpret
   / "CONTENT"
      ( "FIRST"
         ent _ #"FIRSTCONTENT"                                5A8B5A1
      / "NEXT"
         ent _ #"NEXTCONTENT"                                 5A8B5B1
      )
         %display previous content%
         ( IF lastcontent _ xjmpcnt(#"CONTENT") SHOW (
         lastcontent ) RPT dest _ FALSE %for repeat%
            / dest _ LSEL(#"TEXT"))
         vs _ viewspecs()                                     5A8B5C3
      CONFIRM                                                  5A8B5D
      xjump( ent, dest, vs, WINDOW )                           5A8B5E
   / "WORD"
      ( "FIRST"
         ent _ #"FIRSTWORD"                                   5A8B6A1
      / "NEXT"
         ent _ #"NEXTWORD"                                    5A8B6B1
      )
         %display previous content%
         ( IF lastcontent _ xjmpcnt(#"WORD") SHOW (
         lastcontent ) RPT dest _ FALSE %for repeat%
            / dest _ LSEL(#"WORD"))
         vs _ viewspecs()                                     5A8B6C3
      CONFIRM                                                  5A8B6D
      xjump( ent, dest, vs, WINDOW )                           5A8B6E
   );

   jmpcoms =                                                   5A8B8
      ent _                                                    5A8B8A
         ( "ITEM" ent _ #"STATEMENT"
         / "SUCCESSOR"
         / "PREDECESSOR"
         / "UP"
         / "DOWN"
         / "HEAD"
         / "TAIL"
         / "END" <"of Branch">
         / "BACK"
         / "ORIGIN"
         / "NEXT"!L2!
         ) dest _ DSEL(#"CHARACTER")
         vs _ viewspecs()                                     5A8B8A13
      /
         ( ent _ "LINK"
            dest _ LSEL(#"LINK") vs _ NULL                    5A8B8B1A
         / ent _ "NAME"
            ( "ANY"
               ent _ #"NAME"                                  5A8B8B2A1
            / "FIRST"
               ent _ #"FIRSTNAME"                             5A8B8B2B1
            / "NEXT"
               ent _ #"NEXTNAME"                              5A8B8B2C1
            / "EXTERNAL"
```

```
                    ent _ #"EXTNAME"                                5A8B8B2D1
                  )
                    dest _ LSEL(#"NAME") vs _ viewspecs()           5A8B8B2E1
               );

         jmpret =                                                    5A8B9
            ent _ "RETURN"                                           5A8B9A
               CONFIRM                                               5A8B9A1
               dest _ xgetjumpring( ent, WINDOW )  %get statements%
                                                                     5A8B9A2
               pfjmpi(FALSE)  %initialize index for statement array%
                                                                     5A8B9A3
               jmpring
            / ent _ "FILE"
               ( "NAMED"
                  ent _ #"FILENAMED"                                 5A8B9B1A
                  dest _ LSEL(#"OLDFILENAME")                        5A8B9B1B
                  vs _ viewspecs()                                   5A8B9B1C
                  CONFIRM                                            5A8B9B1D
                  xjump( ent, dest, vs, WINDOW )                     5A8B9B1E
               / "RETURN"
                  ent _ #"FILERETURN"                                5A8B9B2A
                  CONFIRM                                            5A8B9B2B
                  dest _ xgetjumpring( ent, WINDOW )  %get file
                  names%                                             5A8B9B2C
                  pfjmpi(FALSE)  %initialize index for file name
                  array%                                             5A8B9B2D
                  jmpring

               );

            jmpring =  %display strings from array until user says
            OK%                                                      5A8B9C
               SHOW (#"
               ")  %go to new line%                                  5A8B9C1
               SHOW (pfjmps(dest))                                   5A8B9C2
               param _ ANSWER                                        5A8B9C3
               (IF param xjumpreturn(ent, pfjmpi(TRUE), WINDOW)
               / IF NOT param jmpring);

   quit COMMAND =  "QUIT"                                            5A9
      quitbody;

      quitbody =                                                     5A9B
         %+BASESUB%                                                 5A9B1
         %-NSW%                                                     5A9B2
         CONFIRM feendsubsys();                                     5A9B3
         %-NSW%                                                     5A9B4
         %+BASESUB%                                                 5A9B5
         %-BASESUB%                                                 5A9B6
         CONFIRM feendsubsys() xquitsubsys() ;                      5A9B7
         %-BASESUB%                                                 5A9B8
   comment COMMAND = ";"                                            5A10
      (IF DISPLAY simtty() param _ TRUE / IF NOT DISPLAY param _
      FALSE)
      cmntpf()                                                      5A10B
```

```
      IF param simdisplay();
                                                                    5A10C
% BASE COMMANDS %                                                     5B
   INITIALIZATION                                                    5B1
      initrule =                                                     5B1A
         seterm %set TERMINATORS%                                    5B1A1
         readsubs _ TRUE                                             5B1A2
         insmode _ FALSE                                             5B1A3
         %-DPS%                                                      5B1A4
         %+NSW%                                                      5B1A5
         xnswinit (pfproject(), pfnode()                             5B1A6
            -> param, param2 %mail msg%, entsubsystem %entry
            subsystem%, param3 %startup commnads branch file%)
         %+NSW%                                                      5B1A7
         %-NSW%                                                      5B1A8
         ent _ feusrid() %  get ident string %                       5B1A9
         xinit [helpusid] ( ent                                     5B1A10
            -> param, param2 %mail msg%, entsubsystem %entry
            subsystem%, param3 %startup commnads branch file%)
         %-NSW%                                                     5B1A11
         [IF NOT param
            <"xinit failed"> feterminate()]
         [IF param2 SHOWSTATUS(param2)]
         ttysim _ xchksimtty() %check if TTY is simulated %         5B1A14
         [IF param3
            feprocess(param3)]
         [IF entsubsystem
            %There is a special entry subsystem.  Go to it%
               dent _ xgoto (TRUE  %name is a string%,           5B1A16A1
               entsubsystem)
               fenewsubsys(dent)]
         %-DPS%                                                     5B1A17
         ;
   REENTRY
      reenrule =                                                     5B2A
         readsubs _ TRUE                                            5B2A1
         seterm
         [IF entsubsystem
            %quit from entry subsystem.  terminate session%
            feendsubsys()];
   %Help rules%                                                      5B3
      %initrule HELP rule - get userid if necessary%                 5B3A
         helpusid =  <"ident = "> param _ LSEL(#"IDENT")            5B3A1
         CONFIRM                                                    5B3A2
         RESUME(param);
   append COMMAND = "APPEND"                                        5B4
      ent _ "STATEMENT"                                             5B4A
      <"at"> source _ SSEL(#"STATEMENT") <"to">
      dest _ DSEL(#"STATEMENT")                                     5B4C
      <"join with"> param _ LSEL(#"TEXT")
      CONFIRM                                                       5B4E
      xappend( ent, source, dest, param, 1 );
   break COMMAND = "BREAK"                                          5B5
      (
         ent _ "STATEMENT"                                          5B5A1
            param _ NULL                                           5B5A1A
```

```
            source _ NULL type _ NULL                                   5B5A1B
            <"at"> dest _ DSEL(#"CHARACTER")
            ( level _ levadj()
               / CONFIRM
               / ( OPTION <"with inserted text">
                  param _ LSEL(#"TEXT")                                 5B5A1D2A
                  ( level _ levadj()
                  / CONFIRM )
                  )
            )
         / IF DISPLAY ent _ "WINDOW"
            level _ NULL                                                5B5A2A
            (
               type _ "VERTICALLY"                                      5B5A2B1
               / type _ "HORIZONTALLY"
            )
            param _ NULL                                                5B5A2D
            <"at">
               [ param _ "CENTER" <"of window">]
            source _ DSEL(#"WINDOW")                                    5B5A2F
            <"displaying in"> dest _ DSEL(#"WINDOW")
            CONFIRM
      )
      xbreak( ent, source, dest, level, param, type, FALSE );
   bslash COMMAND = IF TYPEWRITER "\"                                   5B6
      xprtstmt( WINDOW -> shwstr )
       SHOW ( shwstr ) ;
   clear COMMAND = IF DISPLAY "CLEAR"!L2!                               5B7
      <"status window"> CONFIRM pfclrwnd(1002 %tty window%);
   %-NSW%                                                               5B8
   connect COMMAND = "CONNECT"!L2!                                      5B9
      <"to">
      %
      (ent _ ("DISPLAY"!L2! / "TTY")
         <"Number"> dest _ LSEL(#"NUMBER") <"for">
         param _ ("INPUT"! 1! <"and Output"> / "OUTPUT" <"Only">) /
      %
      ent _ "DIRECTORY"  dest _ LSEL(#"DIRECTORY")
         param _ NULL <"password">
         [ param _ LSEL(#"PASSWORD")  ]

      CONFIRM
      shwstr _ xconnect(ent, dest, param)
      IF shwstr SHOW(shwstr);  % should be SHOWSTATUS when FE allows
      it %                                                              5B9H
   %-NSW%                                                               5B10
   copy COMMAND = "COPY"                                                5B11
      vs _ NULL
      level _ NULL
      source _ NULL
      dest _ NULL
      ( type _ ("LINK" / "NUMBER")
         copy1
         dent _ #"VISIBLE"
         dest _ DSEL(dent)
      / type _ text1
```

```
         copy1
         dent _ type
         dest _ DSEL(dent)
      / type _ "TEXT"
         copy1
         dent _ #"CHARACTER"
         dest _ DSEL(dent)
      / type _ structure
         copy1
         dent _ #"STATEMENT"
         dest _ DSEL(dent)
         [OPTION <"filtered:"> vs _ viewspecs()]
         level _ levadj()
      / type _ "SEQUENTIAL"!L2!
         <"file from"> source _ LSEL(#"OLDFILENAME")
         <"to follow"> dest _ DSEL(#"STATEMENT")
         level _ levadj()
         CLEAR <"using">
         (dent _ "ONE" <"<CR> to end statement">
         / dent _ "TWO"  <"<CR>s end statement">
            [ dent _ "JUSTIFIED"  <"delete extra <SP>"> ]
         / dent _ "ASSEMBLER"
         )
      %-NSW%                                                        5B11J
      / type _ "DIRECTORY"
         dent _ NULL
         copy3
         [diropt]
      %-NSW%                                                        5B11L
      / type _ "FILE"
         dent _ type
         <"from"> source _ LSEL(#"OLDFILENAME")
         <"to a new file to be named"> dest _ LSEL(#"NEWFILENAME")
      )
      CONFIRM
      xcopy( type, source, dent, dest, level, vs, FALSE, FALSE ->
      shwstr )
      IF shwstr SHOWCONFIRM(shwstr);                                5B11Q
   copy1 = <"from"> source _ SSEL(type) <"to follow">;               5B12
   copy3 =                                                          5B13
      <"of">
      (CONFIRM /  source _ LSEL(#"DIRECTORY"))
      <"to follow"> dest _ DSEL(#"STATEMENT")
      level _ levadj()                                              5B13D
      ;
   create COMMAND = "CREATE"!L2! "FILE"                             5B14
      namfil _ LSEL(#"NEWFILENAME") CONFIRM                         5B14A
      xcreate( namfil, WINDOW -> shwstr );
   delete COMMAND = "DELETE"                                        5B15
      dest _ NULL vs _ NULL source _ NULL
      (
         type _ textent
            <"at"> dest _ DSEL(type)
         / type _ structure
            <"at"> dest _ DSEL(type)
            [ OPTION  <"Filtered:">  vs _ viewspecs() ]
```

```
      / type _ "MARKER"!L2! <"named"> <"not implemented">
         dest _ LSEL(type)
      / type _ "ALL" <"markers"> <"not implemented">
      / type _ "MODIFICATIONS" <"to file">
         CONFIRM <"really?">
      / type _ "FILE"
         dest _ LSEL(#"OLDFILENAME")
      / IF DISPLAY type _ "WINDOW"!L2!
         source _ DSEL(#"WINDOW")                                  5B15B7A
         <"enlarging window"> dest _ DSEL(#"WINDOW" )
   )
   CONFIRM
   shwstr _ xdelete( type, source, dest, vs, WINDOW, FALSE )
                                                                   5B15E
   IF shwstr SHOWCONFIRM(shwstr);                                  5B15F
edit COMMAND = IF TYPEWRITER "EDIT"!L2! "STATEMENT" <"at">          5B16
   dest _ DSEL(#"STATEMENT")
   param _ xgettext( dest, 0 ) param _ editstring(param)           5B16B
   xreplace( dest, param, FALSE, FALSE )
   param _ NULL;                                                   5B16D
enlarge COMMAND = IF DISPLAY "ENLARGE"!L2!                         5B17
   <"window"> source _ DSEL(#"WINDOW")
   <"to"> dest _ DSEL(#"WINDOW")
   CONFIRM
   xexpand(source, dest);
%-NSW%                                                             5B18
expunge COMMAND =                                                  5B19
   "EXPUNGE"!L2! <"deleted files from">
      ent _ "DIRECTORY"
      CONFIRM
      xexpunge(ent);
%-NSW%                                                             5B20
force COMMAND = "FORCE" <"Case">                                   5B21
   ( param _ editentity
      <"at"> dest _ DSEL( param )
      param2 _ NULL  [param2 _ cshmode]
   / param _ "MODE" dest _ NULL
      param2 _ cshmode
   )
   CONFIRM                                                         5B21D
   xforce( param, param2, dest ) ;
   % CASE SHIFT MODES %
      cshmode =                                                  5B21F1
         ( "UPPER"
         / "LOWER"
         / "FIRST" <"letter upper"> );
freeze COMMAND = IF DISPLAY "FREEZE"!L2! "STATEMENT" <"at">        5B22
   dest _ DSEL(#"STATEMENT")
   vs _ viewspecs() CONFIRM
   xfreeze(dest, vs);
insert COMMAND = "INSERT"                                          5B23
   level _ NULL
   (
      (( type _ "LINK"
         <"to follow">
         dest _ DSEL(#"VISIBLE")
```

```
            source _ LSEL(type)
         / type _ "NUMBER"
            <"to follow">
            dest _ DSEL(#"VISIBLE")
            source _ LSEL(type)
         / type _ text1
            <"to follow">
            dest _ DSEL(type)
            source _ LSEL(type)
         / type _ "TEXT"
            <"to follow">
            dest _ DSEL(#"CHARACTER")
            source _ LSEL(type)
         / type _ structure
            <"to follow">
            dest _ DSEL(#"STATEMENT")
            [ level _ levadj() ]
            source _ LSEL(type)
         / type _ "DATE"
            <"to follow">
            dest _ DSEL(#"VISIBLE")
            source _ NULL
         / type _ "TIME"!L2!
            <"and date to follow">
            dest _ DSEL(#"VISIBLE")
            source _ NULL
         / type _ "SENDMAIL"!L2! <"form">
            <"to follow">
            dest _ DSEL(#"STATEMENT")
            level _ levadj()
            source _ NULL
         )
         CONFIRM
         xinsert( type, dest, level, source, FALSE, FALSE ))
      %+PROGRAMMERS%                                              5B23C
      / insprog
      %+PROGRAMMERS%                                              5B23E
      );
   insstatement COMMAND = % INS STATEMENT FUNCTION %               5B24
      ( "<^E>"
      / IF #"<^E>" = TERMCHAR
      / IF insmode
      )
      insstmt;

      insstmt =                                                   5B24G
         pfinss($TERMCHAR, insmode)
         level _ NULL                                             5B24G2
         CLEAR
         [ level _ levadj() ]
         param _ LSEL(#"STATEMENT")                               5B24G5
         CONFIRM xinsstatement( level, param, FALSE, FALSE );     5B24G6
   linefeed COMMAND = IF TYPEWRITER "<LF>" xprtnext(WINDOW ->
   shwstr)                                                        5B25
       SHOW ( shwstr ) ;
   load COMMAND =
```

```
      "LOAD" "FILE"
      namfil _ LSEL(#"OLDFILENAME") CONFIRM                          5B26B
      xopen(namfil, TRUE, FALSE, WINDOW);
   %-NSW%                                                            5B27
   logout COMMAND =                                                  5B28
      "LOGOUT"!L2!  CONFIRM
         xlogout()
         felogout();
   %-NSW%                                                            5B29
   mark COMMAND = "MARK"!L2! "CHARACTER" <"at">                      5B30
      dest _ DSEL( #"CHARACTER" )
      <"with marker named"> source _ LSEL(#"WORD")
      CONFIRM
      xmark( dest, source);
   % Commented out because it never worked                          5B31
   merge COMMAND = "MERGE"!L2!                                       5B32
      type _ notstatement
      <"at"> source _ SSEL(type)
      <"into"> dest _ DSEL(type)
      CONFIRM
      xmerge( source, dest, WINDOW);
   %
   move COMMAND = "MOVE"                                             5B34
      vs _ NULL
      level _ NULL
      source _ NULL                                                  5B34C
      dtype _ NULL                                                   5B34D
      ( type _ "LINK"
         copy1
         dtype _ type
         dest _ DSEL(#"VISIBLE")
      / type _ "NUMBER"
         copy1
         dtype _ type
         dest _ DSEL(#"VISIBLE")
      / type _ text1
         copy1
         dtype _ type
         dest _ DSEL(dtype)
      / type _ "TEXT"
         copy1
         dtype _ #"CHARACTER"
         dest _ DSEL(dtype)
      / type _ structure
         copy1
         dtype _ #"STATEMENT"
         dest _ DSEL(dtype)
         [ OPTION <"filtered:"> vs _ viewspecs()]
         level _ levadj()
      )
      CONFIRM
      xmove(type, source, dtype, dest, level, vs, FALSE, FALSE);
   output COMMAND = "OUTPUT" <"to">                                  5B35
      namfil _ NULL
      param _ NULL % use default number of copies %
      param2 _ NULL
```

```
param3 _ NULL
opttysim _ FALSE                                                    5B35E
( ( type _  "PRINTER"
   [ "APPEND" <"to file">
      namfil _ LSEL(#"OLDFILENAME")
      param2 _ TRUE
   / "FILE" namfil _ LSEL(#"NEWFILENAME")
   / "COPIES" param _ LSEL(#"INTEGER")
   ]
   CONFIRM
   xoutput( type, namfil, WINDOW, param, param2 ))
/ ( type _  "COM"
   <"for device">
   (CONFIRM param4 _ #"COMP80"
      / (param4 _
         ( "SINGER"
         / "500VIDEOCOMP"
         / "800VIDEOCOMP"
         / "COMP80"
         ) )
      )
   [ "APPEND" <"to file">
      namfil _ LSEL(#"OLDFILENAME")
      param2 _ TRUE
   / "FILE" namfil _ LSEL(#"NEWFILENAME")
   / "COPIES" param _ LSEL(#"INTEGER")
   / "TEST"
      param3 _ TRUE
      [ "FILE" namfil _ LSEL(#"NEWFILENAME")
      / "TERMINAL" type _ #"COMTSTTTY"
         ( IF DISPLAY  %simulate typewriter mode%
            xsimulate( #"TYPEWRITER" )
            simtty()
            opttysim _ TRUE
         / IF NOT DISPLAY )
      ]
      CONFIRM % so file will have a confirm in parallel %
   ]
   CONFIRM
   xoutput( type, namfil, WINDOW, param, param2, param3,
   param4 )
   ( IF opttysim  %go back to display mode%
      simdisplay()
      xsimulate( #"DISPLAY" )
      opttysim _ FALSE
   / IF NOT opttysim )
   )
/ ( type _  "QUICKPRINT"
   [ "NO" <"headers"> param3 _ TRUE]
   [ "APPEND" <"to file">
      namfil _ LSEL(#"OLDFILENAME")
      param2 _ TRUE % append. %
   / "FILE" namfil _ LSEL(#"NEWFILENAME")
   / "COPIES" param _ LSEL(#"INTEGER")
   ]
   CONFIRM
```

```
      xoutput( type, namfil, WINDOW, param, param2, param3 ))
    %-NSW%                                                          5B35I
    / ( type _  "JOURNAL" <"quickprint">
       [ "NO" <"headers"> param3 _ TRUE]
       [ "APPEND" <"to file">
          namfil _ LSEL(#"OLDFILENAME")
          param2 _ TRUE % append. %
       / "FILE" namfil _ LSEL(#"NEWFILENAME")
       / "COPIES" param _ LSEL(#"INTEGER")
       ]
       CONFIRM
       xoutput( type, namfil, WINDOW, param, param2, param3 ))
    %-NSW%                                                          5B35K
    / ( type _  ("SEQUENTIAL" /  "ASSEMBLER" )
       ( "APPEND" <"to file">
          namfil _ LSEL(#"OLDFILENAME")
          param2 _ TRUE
       / "FILE"
          namfil _ LSEL(#"NEWFILENAME")
       )
       [ "FORCE" <"upper case"> param3 _ TRUE ]
       CONFIRM
       xoutput( type, namfil, WINDOW, param, param2, param3 ))
    / ( % Terminal or remote %
       ( type _ "TERMINAL" param _ NULL  port _ NULL tfil _ FALSE
       termsim _ TRUE
          [ "FILE" param _ LSEL(#"NEWFILENAME") tfil _ TRUE
          termsim _ FALSE]
       / type _ "REMOTE"
          tfil _ FALSE termsim _ FALSE
          <"printer -- TIP"> param _ LSEL(#"VISIBLE")
          <"port #"> port _ LSEL(#"INTEGER") )
       CONFIRM
       CLEAR
       <"send form feeds?"> ff _ ANSWER
          (IF ff  sim _ FALSE
          /IF NOT ff <"simulate?"> sim _ ANSWER)
       CLEAR
          (IF  NOT tfil <"wait at page break?"> pb _ ANSWER
          /IF  tfil)
       CLEAR
       <"CONFIRM when ready"> mustconfirm()
       (IF termsim
          ( IF DISPLAY  %simulate typewriter mode%
             xsimulate( #"TYPEWRITER" )
             simtty()
             opttysim _ TRUE
          / IF NOT DISPLAY )
       /IF NOT termsim)
       xoutput( type, param, WINDOW, port, ff, sim, pb)
       ( IF opttysim  %go back to display mode%
          simdisplay()
          xsimulate( #"DISPLAY" )
          opttysim _ FALSE
       / IF NOT opttysim )
       )
```

```
    );
period COMMAND = IF TYPEWRITER "."                                  5B36
    xprtsnum( -> shwstr ) SHOW ( shwstr ) ;
print COMMAND =                                                     5B37
    IF TYPEWRITER "PRINT" dest _ NULL vs _ NULL                     5B37A
        ( type _ "REST" CONFIRM
        / type _ "FILE" CONFIRM
        / type _ structure
            <"at"> dest _ DSEL( type )
            vs _ viewspecs()
        %-NSW%                                                      5B37A4
        / type _ "JOURNAL" <"mail">
        %-NSW%                                                      5B37A6
        CONFIRM                                                     5B37A7
        )
        xprint( type, dest, vs, WINDOW)% -> shwstr )
            this used to SHOW return and/or show coroutine returns
                SHOW ( shwstr )%;
%-NSW%                                                              5B38
process COMMAND =                                                   5B39
    (IF DISPLAY "PROCESS" / IF NOT DISPLAY "PROCESS" !L2! )
    <"commands from"> dent _ structure
    <"at"> dent _ DSEL(dent)
    CONFIRM
    xprocess( dent -> sent)
    feprocess(sent);                                                5B39F
%-NSW%                                                              5B40
release COMMAND =                                                   5B41
    IF DISPLAY "RELEASE"!L2!                                        5B41A
        (type _ "FROZEN" <"statement at">
            dest _ DSEL(#"STATEMENT")
        /type _ "ALL" <"frozen statements"> dest _ NULL )
        CONFIRM
        xrelease( type, dest );
renumber COMMAND =                                                  5B42
    "RENUMBER"!L2! "SIDS"
        <"in file"> CONFIRM
        xrenumber(WINDOW );
replace COMMAND =                                                   5B43
    "REPLACE"
        type _ editentity
        <"at"> dest _ DSEL(type)
        <"by"> source _ LSEL(type)
        CONFIRM
        xreplace( dest, source, FALSE, FALSE );
reset COMMAND =                                                     5B44
    "RESET"!L2!
        dest _ NULL
        (
            ((type _ "CASE" <"mode">
            /type _ "CONTENT"!L2! <"pattern">
            /type _ "LINK" <"default for file">
            /type _ "NAME" <"delimiters in">
                param _ structure
                <"at"> dest _ DSEL(param)
            /type _ "TEMPORARY" <"modifications for file">
```

```
            / type _ "VIEWSPECS")
              CONFIRM
              )
           / (IF DISPLAY type _ "TTY"!L2! <"window"> CONFIRM rstty())
           )
           xreset( type, dest, WINDOW );
   set COMMAND =                                                    5B45
      "SET"!L2!
      dest _ NULL  param _ NULL  param2 _ NULL param3 _ NULL         5B45B
      (
         type _ "EXTERNAL" !L2! <"names link file to:">
            param2 _ LSEL(#"LINK")  param _ WINDOW
         / type _ "CONTENT" <"pattern to">
            param2 _ LSEL( #"TEXT" ) param3 _ WINDOW
         / type _ "LINK"
            <"default for file to directory">
            param2 _ LSEL(#"WORD")  param _ WINDOW
         / type _ "NAME"
            <"delimiters in"> param _ structure
            <"at"> dest _ DSEL( param )
            CLEAR
            <"left delimiter"> param2 _ LSEL(#"CHARACTER")
            <"right delimiter"> param3 _ LSEL(#"CHARACTER")
         / "NLS" !L2! <"protection for file">
            type _ ( "PRIVATE" / "PUBLIC"!L2! )
         / type _ "TEMPORARY" <"modifications for file">
            CONFIRM <"really?">  param _ WINDOW
         %-NSW%                                                     5B45C7
         / type _ "EXECUTIVE" <"protection for file named">
            param _ LSEL(#"OLDFILENAME")                            5B45C8A
               param2 _ NULL                                        5B45C8A1
               param3 _ NULL                                        5B45C8A2
               ( "SET" <"to">
                  param2 :_ #"SET"
                  param2 :_ LSEL(#"TEXT")
               / "ALLOW"
                  param2 :_ #"ALLOW"
                  prtgrp
                  prot1
               / "FORBID"
                  param2 :_ #"FORBID"
                  prtgrp
                  prot1
               / "RESET"
                  param2 :_ #"RESET"
               / "PRIVATE"
                  param2 :_ #"PRIVATE"
                  <"for"> prtgrp
               )
         %-NSW%                                                     5B45C9
         / IF DISPLAY type _ "TTY"!L2! <"simulation for window">
            param _ DSEL(#"WINDOW") param _ WINDOW CONFIRM
            setty( param )
         / type _ "VIEWSPECS"
            param _ viewspecs()
         )
```

```
      CONFIRM                                                          5B45D
      xset( type, param, param2, param3,  dest ) ;
      prtgrp =
         param2 :_
            ( "SELF"
            / "GROUP"
            / "PUBLIC"
            );
      prot1 =                                                          5B45G
         ( param2 :_ "SET" <"to">
            param2 :_ LSEL(#"INTEGER")
         /
            ( param2 :_
               ( "READ"
               / "WRITE"
               / "EXECUTE"
               / "APPEND"!L2!
               / "LIST"
               / "ALL"
               ) <"access">
            )
         )
         CLEAR <"finished?">                                           5B45G4
         param4 _ ANSWER                                               5B45G5
         (IF param4 CONFIRM / IF NOT param4 prot1);
   show COMMAND =                                                       5B46
      "SHOW"!L2!
      param _ NULL                                                     5B46B
      (
         (
            ( type _ "FILE"
               param _
                  ( "STATUS"
                  / "DEFAULT" <"directory for links">
                  / "MODIFICATIONS" <"status">
                  / "RETURN" <"ring">
                  / "SIZE"!L2! )
            / type _ "RETURN" <"ring">
            / "MARKER" <"list">
               param _ #"MARKER" type _ #"FILE"
            / type _ "NAME" <"delimiters for statement at">
               param _ DSEL( #"STATEMENT" )
            / type _ "VIEWSPECS" <"status">
               [param _ "VERBOSE"]
            %-NSW%                                                     5B46C1F
            / type _ "DISK"!L2! <"space status">
            %-NSW%                                                     5B46C1H
            )
            CONFIRM
         )
         %-NSW%                                                        5B46C3
         / type _ "DIRECTORY"
            <"of">
            dent _ NULL                                              5B46C4B
            (CONFIRM / param _  LSEL(#"DIRECTORY"))
            (CONFIRM / diropt)
```

```
      %-NSW%                                                    5B46C5
   )
   CLEAR
   xshow( type, param, dent, WINDOW -> shwstr )
   SHOWCONFIRM(shwstr);                                         5B46G
   % DIRECTORY OPTIONS %
      dropt =                                                   5B46H1
         CLEAR
         param2 _ NULL                                          5B46H1B
         ( "ALL"!L2! <"files">
            param2 :_ #"BOTH"                                   5B46H1C1
         / "DELETE"!L2! <"files only">
            param2 :_ #"DELETE"                                 5B46H1D1
         / "UNDELETE" <"files only">
            param2 :_ #"UNDELETE"                               5B46H1E1
         / "FOR" <"file">
            param2 :_ #"FOR"                                    5B46H1F1
            param2 :_ LSEL(#"OLDFILENAME")                      5B46H1F2
         / "ARCHIVE"
            param2 :_ #"ARCHIVE"                                5B46H1G1
            ( "STATUS"
               param2 :_ #"STATUS"                              5B46H1G2A
            / "TAPE" <"numbers">
               param2 :_ #"TAPE"                                5B46H1G3A
            )
         / "ACCOUNT"!L2!
            param2 :_ #"ACCOUNT"                                5B46H1H1
         / "DATE" <"of">
            param2 :_ #"DATE"                                   5B46H1I1
            ( "ARCHIVE"
               param2 :_ #"ARCHIVE"                             5B46H1I2A
            / "CREATION"
               param2 :_ #"CREATION"                            5B46H1I3A
            / "LAST" <"dump">
               param2 :_ #"LAST"                                5B46H1I4A
            / "FIRST" <"version creation">
               param2 :_ #"FIRST"                               5B46H1I5A
            / "READ"
               param2 :_ #"READ"                                5B46H1I6A
            / "WRITE"
               param2 :_ #"WRITE"                               5B46H1I7A
            )
         / "DUMP"!L2! <"tape number">
            param2 :_ #"DUMP"                                   5B46H1J1
         / "EVERYTHING"
            param2 :_ #"EVERYTHING"                             5B46H1K1
         / "LAST" <"writer">
            param2 :_ #"LAST"                                   5B46H1L1
         / "LENGTH"!L2! <"and bytesize">
            param2 :_ #"LENGTH"                                 5B46H1M1
         / "MISCELLANEOUS" <"information">
            param2 :_ #"MISCELLANEOUS"                          5B46H1N1
         / "NUMBER" <"of">
            param2 :_ #"NUMBER"                                 5B46H1O1
            ( "VERSIONS" <"to keep">
               param2 :_ #"VERSIONS"                            5B46H1O2A
```

```
            / "ACCESSES"
               param2 :_ #"ACCESSES"                          5B46H1O3A
            )
         / "NO"!L2!
            param2 :_ #"NO"                                    5B46H1P1
            ( "VERSIONS" <"number">
               param2 :_ #"VERSIONS"                          5B46H1P2A
            / "EXTENSION" <"name">
               param2 :_ #"EXTENSION"                         5B46H1P3A
            )
         / "PROTECT"
            param2 :_ #"PROTECT"                               5B46H1Q1
         / "SIZE" <"in pages">
            param2 :_ #"SIZE"                                  5B46H1R1
         / "TIME" <"and date of">
            param2 :_ #"TIME"                                  5B46H1S1
            ( "ARCHIVE"
               param2 :_ #"ARCHIVE"                           5B46H1S2A
            / "CREATION"
               param2 :_ #"CREATION"                          5B46H1S3A
            / "LAST" <"dump">
               param2 :_ #"LAST"                              5B46H1S4A
            / "FIRST" <"version creation">
               param2 :_ #"FIRST"                             5B46H1S5A
            / "READ"
               param2 :_ #"READ"                              5B46H1S6A
            / "WRITE"
               param2 :_ #"WRITE"                             5B46H1S7A
            )
         / "VERBOSE"
            param2 :_ #"VERBOSE"                               5B46H1T1
         / "GROUP" <"by">
            param2 :_ #"GROUP"                                 5B46H1U1
            ( "REVERSE"
               param2 :_ #"REVERSE"                           5B46H1U2A
               grpopt                                         5B46H1U2B
            / grpopt
            )
         / "SORT"!L2! <"by">
            param2 :_ #"SORT"                                  5B46H1V1
            ( "REVERSE"
               param2 :_ #"REVERSE"                           5B46H1V2A
               srtopt                                         5B46H1V2B
            / srtopt
            )
         ) CONFIRM <"finished?">
         param3 _ ANSWER                                       5B46H1X
         dent :_ param2                                        5B46H1Y
         (IF param3 CONFIRM / IF NOT param3 diropt);
      % options for grouping %
      grpopt =
         ( "NO" <"grouping"> param2 :_ #"NO"
         / "ACCOUNT"!L2! param2 :_ #"ACCOUNT"
         / "ARCHIVE" param2 :_ #"ARCHIVE"
            ( "DATE" param2 :_ #"DATE"
            / "STATUS" param2 :_ #"STATUS"
```

```
            / "TAPE" param2 :_ #"TAPE"
            )
         / "CREATION" <"date"> param2 :_ #"CREATION"
         / "DELETE" <"status"> param2 :_ #"DELETE"
         / "DUMP"!L2! param2 :_ #"DUMP" param2 :_ #"DUMP"
            ( "DATE" param2 :_ #"DATE"
            / "TAPE" param2 :_ #"TAPE"
            )
         / "FIRST" <"version creation"> param2 :_ #"FIRST"
         / "LAST" <"writer"> param2 :_ #"LAST"
         / "NUMBER" <"of versions to keep"> param2 :_ #"NUMBER"
         / "PROTECT" param2 :_ #"PROTECT"
         / "READ" <"date"> param2 :_ #"READ"
         / "WRITE" <"Date"> param2 :_ #"WRITE"
         );
      %options for sorting within groups %
      srtopt =
         ( "ACCOUNT"!L2! param2 :_ #"ACCOUNT"
         / "ALPHABETICAL" param2 :_ #"ALPHABETICAL"
         / "ARCHIVE"!L2! param2 :_ #"ARCHIVE"
            ( "TAPE"!L2! param2 :_ #"TAPE"
            / "TIME" <"and date"> param2 :_ #"TIME"
            )
         / "BYTESIZE" param2 :_ #"BYTESIZE"
         / "CREATION" <"time and date"> param2 :_ #"CREATION"
         / "DELETE" <"status"> param2 :_ #"DELETE"
         / "DUMP"!L2! param2 :_ #"DUMP" param2 :_ #"DUMP"
            ( "TAPE"!L2! param2 :_ #"TAPE"
            / "TIME" <"and date"> param2 :_ #"TIME"
            )
         / "LAST"!L2! <"writer"> param2 :_ #"LAST"
         / "LENGTH" <"in bytes"> param2 :_ #"LENGTH"
         / "NUMBER" <"of"> param2 :_ #"NUMBER"
            ( "ACCESSES" param2 :_ #"ACCESSES"
            / "READ" param2 :_ #"READ"
            / "WRITE" param2 :_ #"WRITE"
            / "VERSIONS" <"to keep"> param2 :_ #"VERSIONS"
            )
         / "FIRST" <"version creation"> param2 :_ #"FIRST"
         / "READ" <"time and date"> param2 :_ #"READ"
         / "SIZE" <"in pages"> param2 :_ #"SIZE"
         / "WRITE" <"time and date"> param2 :_ #"WRITE"
         );
   simulate COMMAND =                                              5847
      (
         IF DISPLAY
            "SIMULATE"!L2!
            <"terminal type">
            type _ "TYPEWRITER" CONFIRM
            xsimulate( type )
            simtty()
            ttysim _ TRUE
         / IF NOT DISPLAY IF ttysim
            "SIMULATE"!L2!
            <"terminal type">
            type _ "DISPLAY" CONFIRM
```

```
            simdisplay()
            xsimulate( type )
            ttysim _ FALSE
      );
   slash COMMAND = IF TYPEWRITER "/" xprtcc( -> shwstr )          5B48
      SHOW ( shwstr ) ;                                           5B48A
   sort COMMAND =                                                 5B49
      "SORT"!L2!
         type _ notstatement
         <"at">
         dest _ DSEL( type )
         CONFIRM
         xsort( dest, WINDOW ) ;
   substitute  COMMAND =                                          5B50
      "SUBSTITUTE"
         vs _ NULL  param _ NULL
         type _ textent
         <"in">
            [ OPTION  <"filtered:"> vs _ viewspecs() ]
            dtype _ structure
         <"at"> dest _ DSEL(dtype)
         % collect pairs of entities of type type %
            subst1
         CONFIRM
         xsubstitute( type, dest, param, vs );
         subst1 =                                                 5B50A8
            CLEAR  <"the new"> SHOW(type) param3 _ LSEL(type)
            <"for all occurences of the old"> SHOW(type) param2 _
            LSEL(type)
            param :_ param3  param :_ param2                      5B50A8C
            <"finished?">
               [ "STATUS"
                  CLEAR
                  alltext _ pfsubc(param)  %true if all literals%
                                                                 5B50A8D1B
                  ( IF alltext pfsubs(type, param)
                  / IF NOT alltext
                     xsubresolve(param -> param2)  %go to BE to
                     resolve%
                     pfsubs(type, param2) )
                  CONFIRM CLEAR <"finished?">                    5B50A8D1E
               ]
            param3 _ ANSWER                                      5B50A8E
            (IF param3 / IF NOT param3 subst1);
   tab COMMAND = "    " xrepeatsearch(WINDOW);                    5B51
   transpose  COMMAND =                                           5B52
      "TRANSPOSE"
         type _ editentity
         <"at"> source _ DSEL( type )
         <"and"> dest _ DSEL( type )
         vs _ NULL
            [OPTION <"filtered:"> vs _ viewspecs()]
         CONFIRM
         xtranspose( source, dest, vs, FALSE, FALSE ) ;
   %-NSW%                                                         5B53
   trim COMMAND =                                                 5B54
```

```
      "TRIM"!L2! "DIRECTORY"
         <"no. versions to keep">
         param _ LSEL(#"INTEGER")                                    5B54A2
         CONFIRM <..."really?">
         CONFIRM
         xtrim( param ) ;
   %-NSW%                                                            5B55
   undelete COMMAND =                                                5B56
      "UNDELETE"!L2!
         ( ent _  "FILE"
            param _ LSEL(#"OLDFILENAME")
         / ent _  "MODIFICATIONS" <"to file">
            param _ NULL
         )

         CONFIRM
         xundelete( ent, param ) ;
   uparrow COMMAND = IF TYPEWRITER "^" xprtprev( WINDOW -> shwstr )
                                                                     5B57
      SHOW ( shwstr ) ;                                             5B57A
   update COMMAND = "UPDATE" <"file">                                5B58
      namfil _ NULL
      (
         %-NSW%                                                     5B58B1
         ent _ "NEW"                                                5B58B2
         /
         %-NSW%                                                     5B58B4
         ent _ "OLD" <"version">                                    5B58B5
         / ent _ "COMPACT"
         / ent _ "RENAME" <"to filename">
            namfil _ LSEL( #"NEWFILENAME" ) )
      CONFIRM
      xupdate( ent, namfil, WINDOW %->  shwstr %)
      % SHOW ( shwstr ) waiting for SHOWSTATUS %;
   verify COMMAND = "VERIFY" "FILE" CONFIRM xverify(WINDOW) ;        5B59
% PROGRAMMER COMMANDS AND RULES %                                     5C
   compile COMMAND =                                                 5C1
      "COMPILE"!L2!
         ( "FILE" sent _ #"FILE"
            <"at"> source _ DSEL(#"STATEMENT")
            <"using"> param _ LSEL(#"OLDFILENAME")
            <"to file"> namfil _ LSEL(#"NEWFILENAME")
         / "PROCEDURE" <"at"> sent _ #"PROCEDURE"
            param _ NULL  namfil _ NULL
            source _ DSEL(#"STATEMENT")
         / "L10" <"user program at"> sent _ #"L10"
            param _ NULL  namfil _ NULL
            source _ DSEL(#"STATEMENT")
         / "CONTENT" <"pattern"> sent _ #"CONTENT"
            param _ NULL  namfil _ NULL
            source _ LSEL(#"CHARACTER") )
         CONFIRM
         xpcompile( sent, source, param, namfil );
   load COMMAND =                                                    5C2
      "LOAD"
         ( "FILE"
```

```
            namfil _ LSEL(#"OLDFILENAME") CONFIRM                5C2A1A
            xopen( namfil, TRUE, FALSE, WINDOW )/
         "PROGRAM"
            namfil _ LSEL(#"OLDFILENAME") CONFIRM                5C2A2A
            xload(namfil, WINDOW ) );
   insprog =                                                        5C3
      (type _ "PROGRAMMER'S" !L2!
         <"element">
         ((param _ temptype
            <"to follow">
            dest _ DSEL(#"STATEMENT")                            5C3A2B
            [level _ levadj()])
         / (param _ "COMMENT"
            <"to follow">
            dest _ DSEL(#"WORD")                                 5C3A3B
            source _ LSEL(#"TEXT"))                              5C3A3C
      )
      CONFIRM
      xpinsert( type, dest, level, source, param)
      );
END.
FINISH                                                              7
```

Base subsystem
  Base is the home subsystem in NLS. It has commands that allow you to read, write, and modify information online and print it, among other things.
% Index in <nls,>
% FIRST SEARCHES
  case mode
    (upper or lower case letters) 2A
      to change the case of a STRING or STRUCTURE
     ##<force>##
      to change the default case mode
     ##<force !mode>##
      to reset case mode to the default
     ##<reset !case>##
      CASEMODE (a VARIABLE)
     ##<mode !casemode>##
      to control case in printing
     ##<publication, directive !case>##
  casemode 2B
      upper or lower case letters--commands to change them
     ##<case>##
      CASEMODE (a VARIABLE)
     ##<mode !casemode>##
  character
  ##<nls, information !character>## 2C
  check 2D
      check to be sure file is good
     ##<verify>##
      check on status of things
     ##<show>##
  content
  ##<nls, content>## 2E
  creating... 2F
      files
     ##<create>##
      information
     ##<insert>##
  default
  ##<nls, default>## 2G
  directory
  ##<nls, directory>## 2H
  disk
  ##<nls, directory !pages>## 2I
  display
  ##<nls, display>## 2J
  duplicate 2K
      duplicating information with the Insert commands
     ##<insert>##
      Copy commands
     ##<copy>##
      printing copies of files
     ##<output>##
  duplication
  ##<duplicate>## 2L
  edit 2M
      to modify (or edit) and write information in files

##<writing>##
Base commands for modifying (or editing) information
##<modifying-commands>##
the TNLS Edit Statement command
##<modifying-commands !edit>##
editing 2N
modifying (or editing) and writing information in files
##<writing>##
Base commands for modifying (or editing) information
##modifying-commands>##
file
##<nls, file>## 2O
frozen 2P
to freeze a statement (keep it on your screen)
##<freeze>##
Release Frozen statement
##<release !frozen>##
viewspec o for turning frozen statement on
##<nls, o>##
input
##<nls, input>## 2Q
inserting commands
##<inserting-commands>## 2R
invisible
##<nls, invisible>## 2S
Journal
##<sendmail, journal>## 2T
line-feed
##<linefeed>## 2U
link
##<nls, link>## 2V
lower case
##<case>## 2W
lowercase
##<case>## 2X
marker
##<nls, marker>## 2Y
modifications
##<nls, modification>## 2Z
modify 2A@
to modify (or edit) and write information in files
##<writing>##
Base commands for modifying (or editing) information
##<modifying-commands>##
Modify subsystem
##<modify,>##
modifying 2AA
modifying and writing information in files
##<writing>##
Base commands for modifying (or editing) information
##<modifying-commands>##
Modify subsystem
##<modify,>##
name
##<nls, name>## 2AB
nls

##<nls, nls>## 2AC
OK
##<nls, ok>## 2AD
partition the display screen into different windows with the command
##<break !window>## 2AE
play
##<playback>## 2AF
private
##<nls, private>## 2AG
printing
##<nls, printing>## 2AH
read
##<reading>## 2AI
recording
##<special-purpose>## 2AJ
return
##<nls, return>## 2AK
seeing information
##<reading>## 2AL
Sendmail 2AM
    Sendmail subsystem
   ##<sendmail, sendmail>##
   form
   ##<insert !sendmail !forms>## 2AM2
    Insert Sendmail (form) command
   ##<insert !sendmail>##
seqtype
##<copy !sequential>## 2AN
sequential
##<nls, sequential>## 2AO
sorting
##<sort>## 2AP
special purpose commands
##<special-purpose>## 2AQ
split the display screen into different windows with
##<break !window>## 2AR
statement
##<nls, statement>## 2AS
status 2AT
    status-commands in Base
   ##<status-commands>##
    Show File Status command
   ##<show !file !status>##
    Show Disk (space status) command
   ##<show !disk>##
stop
##<nls, stop>## 2AU
string
##<nls, string>## 2AV
structure
##<nls, structure>## 2AW
Tenex
##<nls, executive>## 2AX
terminal
##<nls, terminal>## 2AY
text

##<nls, text>## 2AZ
TNLS
##<nls, typewriter>## 2B0
tty
##<nls, status>## 2BA
updating
##<update>## 2BB
upper case
##<case>## 2BC
uppercase
##<case>## 2BD
viewing 2BE
viewing in DNLS
##<reading !viewing>##
commands for viewing
##<viewing-commands>##
viewspecs
##<nls, viewspecs>## 2BF
visible
##<nls, visible>## 2BG
word
##<nls, word>## 2BH

how to use the Base subsystem

You use the Base subsystem by issuing commands. You type in commands, which are "verbs" such as Insert and Delete, usually followed by "nouns" such as Statement and Word. Base commands allow you to perform all the functions listed below. See command (for a description of how to command in NLS). See also: NLS. 3

reading and viewing information

You can read any NLS file whose name you know, except a file whose access has been specifically restricted. You call files with the Jump Link command. After you have loaded a file you can move around within its structure by "pointing" to the specific place you want to go. You can view a file at your terminal in different ways with viewspecs or you can print it out for offline reading. You can read about NLS by using the Help command. See pointing, printing. See also: information. 3A

accessing files

going to a file to read it or write on it. Existing files can be accessed with a FILEADDRESS wherever the prompt A: (ADDRESS) appears. You can also use the Jump Link command to open a file for read or write access. By pointing to a link with the mouse or by typing it in, you can get to see the file it specifies. A record of the files you have seen in during your current NLS session, the file return ring, provides another way of reaching those files easily with the Jump File Return command. When you use the Create File command in NLS, the new file is immediately loaded for you. Access to files may be protected. 3A1

file return ring
##<nls, file !return>##
pointing (moving to a specific character)
##<nls, pointing>##
Jump to Link command
##<nls, jump !link>##
showing lists of files
##<base !show !directory>##

privacy provisions
To use the Executive System (TENEX or TOPS-20), you must know a (secret) password. (NLS recognizes you by an independent IDENT associated with you as a user.) Unless a user has protected her or his files, you may show a list of, read, and execute the program files of another. In addition to these three functions, you may write on your own files and possibly those of a group of co-workers. With the Set Executive (protection) command, you may extend or restrict any of these four functions, to yourself, your group, or all users, for your files. With the Set Nls (protection for file) command you can restrict, to any list of idents, read (and therefore write) access to your files. There are also Executive System commands to protect files (request the "TENEX Guide for Users of NLS" from FEEDBACK), and a Private command in the Sendmail subsystem. 3A1E

Sendmail Private command
##<Sendmail, Sendmail !private>##
write access to files
the capability of writing on a file. Whether or not other people initially have the capability to write on one of your files varies with the NLS system. At Office-1, initially only you can write on your files. If you want to open files to others for writing, you must use the command Set Nls (protection for files) Public. You may ask the operator to open all files in your directory to a group of co-workers or to all users. In that case, you can protect a particular file with the command Set Nls (protection for file) Private. See also: writing, privacy, Calculator Write. 3A1E2
executing program files
##<programs, running>## 3A1E3
% backlinks: <sendmail, private>

moving around in files and printing on your typewriter terminal
The Jump and Print command families are used to view information from a typewriter terminal. Print is the most useful command for moving around in files. Jump to Address is the basic pointing (i.e., specifying a character) command. See also: pointing, file, structural. 3A2

printing at your terminal
In TNLS you can print information at your terminal through two groups of commands which allow you more or less formatting. 1) The Print commands allow you to print as little as a statement or as much as a whole file at your terminal. Print commands allow pagination and simple formatting controlled by viewspecs. 2) The Output Printer commands enable you to use over two hundred directives that give you essentially a publisher's control of layout. Read the Output Processor Users' Guide. See also: printer, reading, pointing. 3A2A

The view you get of a file is controlled by viewspecs.
You can change the kind of view you want by manipulating these:
##<nls, viewspecs>##
Jump commands available in all subsystems
##<nls, jump>##

viewing contents of files on a display screen
When you enter the Base subsystem, your initial file will appear in your file display area, starting with the origin statement and displaying as much as will fit. Every time you go to a new location (by creating a file or by jumping), the display will start at that location and display as much of the text as fits. When you have a particular view on the screen, you can operate upon the text in view by pointing with BUG or by giving an ADDRESS within commands. You can operate on the text not in view by giving an ADDRESS. A new window can be made by using Break Window. You get around in files, i.e., change your view, by jumping. See reading, viewing-commands, initial file, origin statement. See also: accessing pointing, display BUG. 3A3

jumping commands to see information in display mode
There is a whole family of Jump commands, available in all subsystems, to use for moving to a new place in a file. Some Jump commands take you to a character within a statement; some take you to files; and some take you to statements according to their structural position. In display mode, all of them move you to the first character of the statement or of the origin statement if just a file is addressed. If you have created more than one window, the view you specify in a Jump command will appear in the file display window your cursor is in when you give the final OK for the command, once your screen has been re-created. See also: viewing, structural, pointing. 3A3A

Jump commands available in all subsystems
##<nls, jump>##
re-creating the display
##<re-creating>##
jumping in TNLS
##<moving>##

too-big: seeing a statement too big to fit on the screen
There are three ways of seeing text to big to fit in the text display area. 3A3B

Simulate Typewriter
Use the "Simulate Typewriter" command and then the "Print Statement" command. Use "Simulate Display" to get back to DNLS.

Break the statement in two. If you don't have write access to the file, you can "Set Temporary" modifications but there is a bug with "Reset Temporary" that requires manually deleting the modification file created.

JC: Simulate a "Jump (to) Character" command with the following incantation:

ep sen<OK>
<CTRL-H>sldpyarea.daccnt_1000<OK>
c<OK>

Finally, recreate the screen with viewspec F.
You can leave out the first line of the incantation after you have done it once in a session. You can replace the number 1000 with the number of any character you want to have at the top of your screen. The screen may do things you've never seen before but it's fun to watch.
You might send a note to FEEDBACK requesting a "Jump (to) Character" command to do all this for you. 3A3B3

The view you get of a place is controlled by viewspecs.
You can change the kind of view you want by manipulating these
##<nls, viewspecs>##
beginning statement in display work area and your location
After you Jump (or Delete the statement at the top), the display work area starts with the new statement pointed to as soon as the screen is re-created. The exception to this is when a content filter pattern obscures that statement. In this case, the first statement that passes the filter appears at the top of the screen. See also: content-pattern, filter.
re-creating the display
##<re-creating>##
re-creating the display
After you move to another location in a file, change the type of view, or change something in the part of the file(s) you have in view, DNLS has to re-format the file-display area to show the changes in their proper context. If you have viewspec u on, this happens automatically upon execution of every command. Viewspec v suppresses that automatic re-creation, which saves time during some repetitious operations. (Warning: While viewspec v is on, you may point to a statement other than the one you see on your screen if you do not work from the bottom to the top of the screen.) Independently of either viewspec, re-creation will take place immediately upon specifying viewspec f. See capital-f (for F viewspec). 3A3E
Note--Inputting viewspecs with mouse buttons is not a command...
so put in an f at the end of a series of viewspecs.
Viewspec u will not affect this. See mouse buttons.
%scrolling:
The DNLS Linefeed command is not implemented yet, so see jumping.
hardcopy printing and formatting
##<publication, how>##
pointing (moving to a specific character in a file)
##<nls, pointing>##
Help command
##<nls, help>##
Sendmail reading in Base
##<print !journal>##
% Sendmail reading in Readmail subsystem
##<readmail, readmail>##
privacy provisions
##<privacy>## 3A9
commands for viewing
##<viewing-commands>##
%.PES;.IgLS;
writing, creating, and modifying information
In NLS, you can create new files, copy all or selected parts of existing files to other files, insert text by typing into existing files, and edit existing text. Access for these operations may be protected. See also: information. 3B
Use the Insert command to add, duplicate, or create information.
##<insert>##

creating new files
##<create>##
commands for modifying
##<modifying-commands>##
updating modifications
##<update>## 3B4
correcting errors
To get out of a command you have started, type <CTRL-X>. To backspace and delete one character in a TYPEIN, type <CTRL-A> (or Backspace). To backspace and delete back to the previous word, type <CTRL-W> (or Backspace Word). The commands people use most often to correct errors in text that is already online are Substitute and Replace. Use the Delete Modifications command to remove modifications you have made since your last "update". 3B5
bells
If the system rings the bell on your terminal (or prints "ding-a-ling" on terminals that lack bells) either someone is trying to link to you (in which case you will receive a message on your screen or paper), or you have asked the system to complete a command before giving it enough information. Usually you can go on by typing the next logical character.
See also: connect. 3B5A
Delete Modifications command
##<delete !modifications>##
common error messages and what to do about them
##<nls, error>##
writing access may be temporarily restricted
##<nls, modification !restriction>##
privacy provisions
##<privacy>##
hardcopy printing and formatting
##<publication,> 3C
getting help
##<nls, help>##
how to use NLS
##<nls, how>##
%.Pes;.Igs;
commands in Base
Commands in the Base subsystem allow you to name, read, write, and modify information online, and output it to hardcopy. They also provide a wide range of file-viewing and file-handling capabilities.
See Commanding, how, subsystem. 4
% The substatements of this branch are here to accomodate problems with the search algorithm.
Archive File CONTENT [(opt:) ARCHIVEOPT] OK
The Base command "Archive File" permits you to store files on tape to save disk space. With this command, you can mark the file you specify for CONTENT to be archived tonight, or never to be archived. Retrieval or archived files normally takes 15 minutes to half an hour. Files unread for a few weeks are archived automatically. Check with your computer's operator to determine the exact date.
ARCHIVEOPT = 4A1
Delete (After Archiving Finished?) ANSWER
The Archive File command option "Delete" undoes the Prevent option. This is the default case. Another ARCHIVEOPT is

expected if you type "n" for ANSWER.
Prevent (Deletion after archiving Finished?) ANSWER
The Archive File command option "Prevent" will mark the file you specified to be archived but leave it online after it is archived. This command will not affect a file that has already been archived or that has just been retrieved from archive. You must create a new version of the file for this command to have effect. Another ARCHIVEOPT is expected if you type "n" for ANSWER.
Not (allowed Finished?) ANSWER
The Archive File command option "Not" will prevent the file you specified from being copied onto tape if it is not read after several weeks. This command will not affect a file that has already been archived or that has just been retrieved from archive. You must create a new version of the file for this command to have effect. Another ARCHIVEOPT is expected if you type "n" for ANSWER.
Deferred (Finished?) ANSWER
The Archive File command option "Deferred" will mark the file you specify for CONTENT to be copied onto tape at the next archive run if it is not read after several weeks. Another ARCHIVEOPT is expected if you type "n" for ANSWER.
Reset (request status Finished?) ANSWER
The Archive File command option "Reset" will mark the file to be archived and deleted if it is not read for several weeks.

systems-commands: universal commands, Logout
The Base commands in this group include universal commands and Logout. See also: terminal-commands, display-commands. 4B
Logout OK
The Base command "Logout" closes up your work on the computer; you do not need to deal with the Executive level if you use NLS Logout. 4B1
TNLS example:
BASE C: Logout OK:
TERMINATED JOB #, USER... 4B1A
universal commands available in all subsystems
##<nls, universal>## 4B2

inserting-commands: Insert (basic command for entering information)
The Base commands in this group allow you to add, duplicate, or create information in a file. Use the Insert Statement command to insert a paragraph, titles, and headings. To enter many statements in a row, use <CTRL-E> (sometimes an INSRT key on your keyboard) to put you in the "enter mode". See also: Create File. 4C
Insert...
The Base command "Insert" allows you to add, create, or duplicate information. See also: <CTRL-E> 4C1
Character: Insert Character (to follow) DESTINATION CONTENT OK
The Base command "Insert Character" adds the character(s) you specify for CONTENT after the character in an existing statement that you point to for the DESTINATION. Afterwards, you are located at the last character you inserted. 4C1A
Date: Insert Date (to follow) DESTINATION OK
The Base command "Insert Date" adds the current date into your file after the visible you point to for the DESTINATION.

Type <CTRL-T> to see the current date and time. 4C1B
Invisible: Insert Invisible (to follow) DESTINATION CONTENT OK
The Base command "Insert Invisible" adds the character(s) you specify for CONTENT after the character in an existing statement that you point to for the DESTINATION. Afterwards, you are located at the last character you inserted. See also: visible. 4C1C
Link: Insert Link (to follow) DESTINATION CONTENT OK
The Base command "Insert Link" adds the characters you specify for CONTENT after the visible in an existing statement that you point to for the DESTINATION. Necessary spaces and link delimiters (angle-brackets) are added automatically. Afterwards, you are located at the last character you inserted. 4C1D
Number: Insert Number (to follow) DESTINATION CONTENT OK
The Base command "Insert Number" adds the character(s) you specify for CONTENT after the visible in an existing statement that you point to for the DESTINATION. Necessary spaces are added automatically. Afterwards, you are located at the last character you inserted. 4C1E
Sendmail: Insert Sendmail (form) (to follow) DESTINATION LEVEL-ADJUST OK
The Base command "Insert Sendmail (form)" inserts a form (See 1, below) listing Sendmail commands into a file as a statement. It is especially useful as a template, enabling you to send several similar items by changing only the parts that are not repeated. Using the Sendmail form also allows for more extensive editing. Use Base text-editing commands to fill out the form. The system fills in the IDENT of the logged-in user for the Author, but you may change it. Commands you leave blank will be ignored. Use the Process (command form) command in the Sendmail subsystem to automatically execute the commands in the form. After insertion you are at the first character of the list of commands. See Sendmail. See also: Sendmail Insert Status. 4C1F

forms: Sendmail forms for all commands
a single statement with a list of Sendmail commands that you may use as a template for Sendmail items. Each command ends with a carriage return in the form statement except the SEND command which must appear at the end and be terminated by a period. Do not use a carriage return in the text of a MESSAGE in the Sendmail form as any carriage return means "end of the message". This means you may not use <CTRL-V> <CR> either. Any of the following commands can be in a Sendmail form. Any commands misspelled, containing lower case letters, or not listed below will be ignored. Many prudent users delete "SEND" (but not the preceding carriage return), and send the item manually after reviewing it by using the Show Status command to check their input. (The menu numbers are not part of the forms.) See Insert Sendmail. See also: Sendmail Process. 4C1F1

TITLE
##<Sendmail,title>##

```
        COMMENT
       ##<Sendmail,comment>##
        AUTHOR(S)
       ##<Sendmail,authors>##
        NUMBER
       ##<Sendmail,number>##
        DISTRIBUTE FOR ACTION TO
       ##<Sendmail,distribute>##
        DISTRIBUTE FOR INFO-ONLY TO
       ##<Sendmail,distribute>##
        SUBCOLLECTION(S)
       ##<Sendmail,subcollections>##
        KEYWORD(S):  TYPEIN
       ##<Sendmail,keywords>##
        HANDLING INSTRUCTION
       ##<Sendmail,expedite>##
        RECORDING INSTRUCTION
       ##<Sendmail,unrecorded>##
        OFFLINE ITEM -- LOCATED AT
       ##<Sendmail,offline>##
        RFC NUMBER
       ##<Sendmail,rfc>##
        OBSOLETES ITEM NUMBER(S)
       ##<Sendmail,obsoletes>##
        ACCESS STATUS
       ##<Sendmail,private>##
        UPDATE TO ITEM NUMBER(S)
       ##<Sendmail,update>##
        INSERT LINK TO FOLLOW
       ##<Sendmail,insert !link>##
        FORWARD ITEM NUMBER
       ##<Sendmail,forward>##
        MESSAGE
       ##<Sendmail,message>##
        STATEMENT AT
       ##<Sendmail,statement>##
        BRANCH AT
       ##<Sendmail,branch>##
        PLEX AT
       ##<Sendmail,plex>##
        GROUP AT
       ##<Sendmail,group>##
        FILE:
       ##<Sendmail, file>##
        SEND THE MAIL
       ##<Sendmail, send>##
       % backlinks:  <sendmail, forms>, <sendmail, process:t>
     Sendmail's Process (command form) command
    ##<Sendmail, process>##
     Base text-editing commands
    ##<modifying-commands>##
    % backlinks:  <sendmail, insert !status.d>, <sendmail,
    process:t>
Statement:  Insert Statement (to follow) DESTINATION
LEVEL-ADJUST CONTENT OK
  The Base command "Insert Statement" allows you to create new
```

statements in your file. To insert many statements in a row, use <CTRL-E> (<ENTER>) to put you in the "enter mode". CONTENT allows the specification of a DESTINATION, i.e. you can also "copy" an existing statement, instead of "inserting" a new one, with this command. After execution you are at the first character of the newly inserted statement. 4C1G

TNLS example:
BASE C: Insert C: Statement (to follow) A: 2b3
L: d
T: How much wood would a woodchuck chuck?
BASE C: 4C1G1
<CTRL-E> (<ENTER>) for the "enter mode"
##<nls, okinsert>##

Text: Insert Text (to follow) DESTINATION CONTENT OK
The Base command "Insert Text" adds the character(s) you specify for CONTENT after the character in an existing statement that you point to for the DESTINATION. Afterwards, you are located at the last character you inserted. See also: Insert Visible, Insert Character. 4C1H

Time: Insert Time (and date to follow) DESTINATION OK
The Base command "Insert Time" writes the current time and date after the visible in an existing statement that you point to you for DESTINATION. Necessary spaces are added automatically. 4C1I

Visible: Insert Visible (to follow) DESTINATION CONTENT OK
The Base command "Insert Visible" adds the character(s) you specify for CONTENT after the visible in an existing statement that you point to for the DESTINATION. Necessary spaces are added automatically. Afterwards, you are located at the last character you inserted. 4C1J

Word: Insert Word (to follow) DESTINATION CONTENT OK
The Base command "Insert Word" adds the character(s) you specify for CONTENT after the word in an existing statement that you point to for the DESTINATION. Necessary spaces are added automatically. Afterwards, you are located at the last character you inserted. 4C1K

STRING: Insert STRING (to follow) DESTINATION CONTENT OK
The Base command "Insert STRING" allows a new STRING to be entered in an existing statement. CONTENT allows you to "copy" an existing STRING rather than insert a new one. After execution you are at the last character of the inserted text. If you insert a link, angle bracket delimiters are automatically supplied if you do not include them in your TYPEIN. See also: Substitute, Replace, link. 4C1L

TNLS example:
BASE C: Insert C: Visible (to follow) A: 3 "9:30"
T: p.m.
BASE C: 4C1L1
STRING = Character, Text, Word, Visible, Invisible, Number, Link
##<nls, string>##
%Nouns.
Character:
Insert Character is a special case of Insert ##<insert !string>## 4C1L3
Word:

```
            Insert Word is a special case of Insert ##<insert
            !string>##                                          4C1L3B
            Visible:
            Insert Visible is a special case of Insert ##<insert
            !string>##                                          4C1L3C
            Invisible:
            Insert Invisible is a special case of Insert ##<insert
            !string>##                                          4C1L3D
            Number:
            Insert Number is a special case of Insert ##<insert
            !string>##                                          4C1L3E
            Link:
            Insert Link is a special case of Insert ##<insert
            !string>##                                          4C1L3F
            Text:
            Insert Text is a special case of Insert ##<insert
            !string>##                                          4C1L3G
         %
            File
            ##<create>##                                        4C1M1
         STRUCTURE:  Insert STRUCTURE (to follow) DESTINATION
         LEVEL-ADJUST CONTENT OK
           The Base command "Insert STRUCTURE" allows you to create new
         statements in your file.  To insert many statements in a row,
         use <CTRL-E> (<ENTER>) to put you in the "enter mode".
         CONTENT allows the specification of a DESTINATION, i.e. you
         can "copy" an existing Statement, Branch, Plex, Group, instead
         of "inserting" a new one, with this command.  If you choose
         TYPEIN with Branch, Plex, or Group, it acts as Insert
         Statement.  After execution, you are at the first character of
         the newly inserted STRUCTURE.                          4C1N
            STRUCTURE = Statement, Branch, Plex, or Group
            ##<nls, structure>##                                4C1N1
            TNLS example:
            BASE C: Insert C: Statement (to follow) A: 2b3
            L: d
            T: Statement is one of four STRUCTURE command words.
            BASE C:                                             4C1N2
             <CTRL-E> (<ENTER>)
            ##<nls, okinsert>##
            %Nouns.
               Statement:
               Insert Statement is a special case of Insert ##<insert
               !structure>##                                    4C1N4A
               Branch:
               Insert Branch is a special case of Insert ##<insert
               !structure>##                                    4C1N4B
               Plex:
               Insert Plex is a special case of Insert ##<insert
               !structure>##                                    4C1N4C
               Group:
               Insert Group is a special case of Insert ##<insert
               !structure>##                                    4C1N4D
       <CTRL-E> (<ENTER>)
      ##<nls, okinsert>##
modifying-commands:  Copy, Delete, Move, Substitute, Update, etc.
```

The Base commands in this group allow you to modify--or edit--STRUCTURES and STRINGS that already exist in files. Some of the commands also allow you to act on whole files. See also: file-handling, writing, Insert. 4D

Append Statement (at) SOURCE (to) DESTINATION (join with) CONTENT OK

The Base command "Append" attaches one statement to another. The appended statement is added to the end of the receiving statement. You may only append statements. Any substructure under the statement to be appended will appear before any substructure under the receiving statement. The characters you specify to "join with" for CONTENT will be inserted between the two statements. (If you're in TNLS and don't want anything inserted, just type a <CR> for CONTENT; for DNLS use <CTRL-N>). After execution you are at the last character of the receiving statement (the character preceding the appended material). 4D1

TNLS example:
BASE C: Append C: Statement (at) A/[T]: 1b11
(to) A: 1b10
(join with) T/[A]: and furthermore
BASE C: 4D1A

Break

Use the Break command to divide one statement or one window into two. 4D2

Statement (at) DESTINATION LEVEL-ADJUST OK

The Base command "Break Statement" divides one statement into two statements. It will break immediately after the next visible following the DESTINATION you point to. You may specify the level of the second statement relative to the first one. 4D2A

TNLS example:
BASE C: Break C: Statement at A: 1 "hobgoblin"
L: d
BASE C: 4D2A1

effects

The second statement will begin with the visible following the one you specified in the command. The invisible that was between the two visibles will disappear. If you want the second statement to be at the same level as the first statement, just type <CR> (or Command Accept) for LEVEL-ADJUST. After execution you are at the first character of the second statement. 4D2A2

Window ...

The Break Window command allows you to divide your screen into separate work areas where you may view different files or different parts of the same file. 4D2B

Vertically: Break Window Vertically (at) SPOT (displaying in) SPOT OK

SPOT = A point on the screen marked by the cursor.

The location of your cursor when you type <OK> determines which side of the screen will contain the information in your old display area. All of your file and statement return memories are located in your old display area. New display areas start over fresh. The new area will say "Empty". Subsequent operations in NLS will take place in one of these areas at a time, whichever one contains your

cursor when you give the final <OK>. The Jump command puts things in the area containing the cursor. Use the Delete Window command to close one window and thus enlarge the window that you bug. Once a window has been deleted, it cannot be retrieved. See also: tty-simulation, screen, Enlarge Window. 4D2B1

Horizontally: Break Window Horizontally (at) SPOT (displaying in) SPOT OK

SPOT = A point on the screen marked by the cursor.

The location of your cursor when you type <OK> determines which side of the screen will contain the information in your old display area. All of your file and statement return memories are located in your old display area. New display areas start over. The new area will say "Empty". Subsequent operations in NLS will take place in one of these areas at a time, whichever one contains your cursor when you give the final <OK>. The Jump command puts things in the area containing the cursor. Use the Delete Window command to close one window and enlarge the window that you bug. Once a window has been deleted, it cannot be retrieved. See also: tty-simulation, screen, Enlarge Window. 4D2B2

Center (of window)

Following the specification of "Horizontally" or "Vertically" in the Break Window you can type the command word "Center". The system will then divide your window evenly in the middle. 4D2B3

empty display areas

The word "Empty" describes an area on the screen that displays nothing. The display area will remain empty until you put something there by simulating teletype, or by using any of the Jump commands. 4D2B4

% backlinks: <nls, edges>, <nls, empty>, <enlarge>, <delete !window>

Copy

The Base command "Copy" allows you to reproduce a SOURCE you specify at the DESTINATION you specify. 4D3

STRING: Copy STRING (from) SOURCE (to follow) DESTINATION OK

The Base command "Copy STRING" reproduces a STRING at another location. SOURCE provides you the choice of TYPEIN. After execution, you are at the last character of the new STRING. 4D3A

TNLS example:

BASE C: Copy C: Word (from) A: 2 "mutual" (to follow) A: 4 "humans"

BASE C: 4D3A1

STRING = Character, Text, Word, Visible, Invisible, Number, or Link

##<nls, string>##

%Nouns.

Character

The Base command "Copy Character" is a special case of Copy ##<copy !string>## 4D3A3A

Word

The Base command "Copy Word" is a special case of Copy ##<copy !string>## 4D3A3B

Visible
The Base command "Copy Visible" is a special case of
Copy ##<copy !string>## 4D3A3C
Invisible
The Base command "Copy Invisible" is a special case of
Copy ##<copy !string>## 4D3A3D
Number
The Base command "Copy Number" is a special case of
Copy ##<copy !string>## 4D3A3E
Link
The Base command "Copy Link" is a special case of Copy
##<copy !string>## 4D3A3F
Text
The Base command "Copy Text" is a special case of Copy
##<copy !string>## 4D3A3G

STRUCTURE: Copy STRUCTURE (from) SOURCE (to follow)
DESTINATION [(Filtered:) FILTER] LEVEL-ADJUST OK
This Base subsystem group of "Copy" commands reproduces the STRUCTURE at another location. SOURCE provides you the choice of TYPEIN. After executing the Copy command, the same STRUCTURE exists in two places, and you are at the first character of the new STRUCTURE. Note--Copy does not pay attention to the viewspecs you have on (i.e., it works throughout the STRUCTURE you point to whether all of it's visible to you or not), unless you specify VIEWSPECS in the FILTER option, which pays attention to level, content analyzer, and sequence generator viewspecs. You cannot limit the effects to certain lines (like first lines only) with this command. 4D3B

TNLS example:
BASE C: Copy C: Branch (from) A: 5d
(to follow) A: 1c
L:
BASE C: 4D3B1
FILTER
##<nls, filter>##
STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.
Statement
The Base command "Copy Statement" is a special case of
Copy ##<copy !structure>## 4D3B4A
Branch
The Base command "Copy Branch" is a special case of
Copy ##<copy !structure>## 4D3B4B
Plex
The Base command "Copy Plex" is a special case of Copy
##<copy !structure>## 4D3B4C
Group
The Base command "Copy Group" is a special case of
Copy ##<copy !structure>## 4D3B4D

Directory: Copy Directory (of) CONTENT (to follow)
DESTINATION LEVEL-ADJUST [DIROPT] OK
The Base command "Copy Directory" copies any directory as a plex following the address you specify for DESTINATION. CONTENT lets you type or point to the name of the directory.

After execution, you are at the first character of the first statement of the copied directory. 4D3C

TNLS example:

BASE C: Copy C: Directory (of) OK/T: MaBell
(to follow) A: 3d
L: d
()OK:
BASE C: 4D3C1
DIROPT option
##<nls, diropt>##

File: Copy File (from) CONTENT (to) CONTENT OK

The Base command "Copy File" copies the file (and its partial copy) from the FILEADDRESS you specify for CONTENT to the new name at the second FILEADDRESS you specify. The command obeys the rules of privacy and protection: you can copy only files you can read, only into directories where you can write. Copy File fails to rewrite the origin statement and any content that existed in the DESTINATION file will not appear in the new version. It will not work when the file to be copied is loaded. Note: Copy File must be used carefully. See effects, below, before using. See also: modification file. 4D3D

TNLS example:

BASE C: Copy C: File (from) T/[A]: pooh (to) T/[A]: tiger
Copied Files Are:
< WEINBERG, POOH.NLS;61, > [and PC] to < WEINBERG,
TIGER.NLS;61, > [and PC]
< WEINBERG, POOH.NLS;60, > to < WEINBERG, TIGER.NLS;60, >
BASE C: 4D3D1
effects

The command produces different results depending on what you type in for CONTENT. The following four cases are typical: 4D3D2

1) When a version number is not specified for either the first or second FILEADDRESS, all versions of the source file are copied, with the new versions keeping the same version numbers as the copied files. If you do not specify the version number of your first file, the command will only work if no version number for the second file is specified. 2) When a new version number is specified in the second FILEADDRESS, the new file has that number regardless of the version number of the first FILEADDRESS. Only one new file is created. 3) When a version number is specified in the first FILEADDRESS, but no number is specified for the second, the new file has the same version number as the original file. 4) When the version number given in the second FILEADDRESS is the same as that of an already existing file by that name, the file will not be copied--the command will not overwrite.

When to use Copy Branch rather than Copy File

Use Copy Branch to copy files without replacing the contents of the second file and creating a new version of that file.

Sequential: Copy Sequential (file from) CONTENT (to follow) DESTINATION LEVEL-ADJUST (using) SEQTYPE OK

The Base command "Copy Sequential" copies the text from the sequential file at the FILEADDRESS you specify for CONTENT into the NLS file at the DESTINATION you specify. The SEQTYPE "Two (<CR>s ends statement)" should always be used when inputting a former NLS file that was "Output Sequentialed" with viewspec y on, and not has not undergone a process that introduces or removes any carriage returns.

SEQTYPE = 4D3E

One (<CR> to end statement) OK

Each line of the sequential file will be a statement, its level determined by the number of spaces or tabs at the beginning of the line. Blank lines will show up as seperate statements containing one space. 4D3E1

Two (<CR>s ends statement) Justified/OK

A new statement will occur after two successive carriage returns. Single carriage returns which occur after 62 characters will be changed into spaces; those before 62 characters will remain carriage returns. The level of a statement is determined by the indentation of the second line of the statement. (The first line may be indented extra spaces to indicate paragraphs.) This is the recommended SEQTYPE. 4D3E2

Justified (delete extra <SP>) OK

This Copy Sequential (file) command will remove multiple spaces from lines that have been right justified in the source file. It will only remove multiple spaces from lines longer than 62 characters. 4D3E2A

Assembler OK

This is for programs written in assembly language. 4D3E3

Delete

The Base command "Delete" erases something you specify such as a character, word, statement, etc. from the DESTINATION you specify. 4D4

STRING: Delete STRING (at) DESTINATION OK

The Base command "Delete STRING" erases the type of STRING you specify from a statement. After execution you are at the first character following the deleted STRING (or at the new last character if the deletion included the end of the statement). 4D4A

TNLS example:

BASE C: Delete C: Character (at) A: 1 +2c

OK:

BASE C: 4D4A1

STRING = Character, Text, Word, Visible, Invisible, Number, Link

##<nls, string>##

%Nouns.

Character

The Base command "Delete Character" is a special case of Delete ##<delete !string>## 4D4A3A

Word

The Base command "Delete Word" is a special case of Delete ##<delete !string>## 4D4A3B

Visible

The Base command "Delete Visible" is a special case of

Delete ##<delete !string>## 4D4A3C
Invisible
The Base command "Delete Invisible" is a special case of Delete ##<delete !string>## 4D4A3D
Number
The Base command "Delete Number" is a special case of Delete ##<delete !string>## 4D4A3E
Link
The Base command "Delete Link" is a special case of Delete ##<delete !string>## 4D4A3F
Text
The Base command "Delete Text" is a special case of Delete ##<delete !string>## 4D4A3G

STRUCTURE: Delete STRUCTURE (at) DESTINATION [(filtered:) FILTER] OK
The Base command "delete STRUCTURE" erases the type of STRUCTURE you specify from a file. You may not use Delete Statement if the statement has substatements. (Instead, try Delete Branch with a FILTER of viewspec e. See filter). After execution, you are at the first character of the next statement following the deleted STRUCTURE (or the new last statement if you deleted the last statement in the file). Note--Delete works throughout the STRUCTURE you point to (whether all of it is visible to you or not), unless you specify VIEWSPECS in the FILTER option. See substructure. 4D4B

TNLS example:
BASE C: Delete C: Branch (at) A: 2
OK:
BASE C: 4D4B1
FILTER option: enter VIEWSPECS:
This option selects statements to be deleted by certain viewspecs you can specify as a string of character codes. Type the OPTION character, <CTRL-U>, first and then enter any combination of the following viewspecs: for level--a,b,c,d,e,w,x; for content analyzer--i,j,k; for sequence generator--O,P. Statements that are left somewhere without a source, because the source passed the FILTER (and thereby got deleted) although they didn't, will be promoted up in level.
STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.
Statement
The Base command "Delete Statement" is a special case of Delete ##<delete !structure>## 4D4B4A
Branch
The Base command "Delete Branch" is a special case of Delete ##<delete !structure>## 4D4B4B
Plex
The Base command "Delete Plex" is a special case of Delete ##<delete !structure>## 4D4B4C
Group
The Base command "Delete Group" is a special case of Delete ##<delete !structure>## 4D4B4D

All: Delete All (markers) OK
The Base command "Delete All (markers)" erases all the

markers in the file you have loaded. If you want to delete only one marker, use the Delete Marker command. To see a list of the markers in your file use the command Show Marker (list). See also: Mark. 4D4C

TNLS example:
BASE C: Delete C: All (markers) OK:
BASE C: 4D4C1

File: Delete File CONTENT OK

The Base command "Delete File" removes the file named for CONTENT from normal use. If you do not name a directory, Delete File assumes you mean a file in your own. Expunge erases forever all deleted files in that directory. Undelete File recalls a deleted file that has not been expunged. Deleted files risk being expunged periodically by the system. Showing deleted files is an option in the Base subsystem Show Directory command. See also: fileaddress. 4D4D

TNLS example:
BASE C: Delete C: File T/[A]: tiger
OK:
Deleted Files Are:
< WEINBERG, TIGER.NLS;61, > and its partial copy
< WEINBERG, TIGER.NLS;60, >
BASE C: 4D4D1

effects

The Delete File command makes the file disappear from the directory list. If you do not specify a version number following the name of the file you typein for CONTENT, all versions of the file and their partial copies will be deleted. Until a file has been expunged it can be undeleted. Occasionally your deleted files will be expunged automatically. The second oldest version of every file is automatically deleted when you create a new version. You may see files in your directory with the extension "PC". Don't delete them. They will be deleted automatically when you use the Update or Delete Modifications commands. See Undelete File. 4D4D2

Expunge Directory command
##<expunge !directory>##
modification file
##<nls, modification>##
Undelete File command
##<undelete !file>##
expunging deleted files
##<expunge>##

effect on file return ring

If a file is on your file return ring and is deleted, it will continue to be echoed as a choice on your return ring, but you will not be able to load the file. If you use the Undelete command before the file is expunged, you will then be able to load it again as if you never deleted it. See also: File Return. 4D4D4

Marker: Delete Marker (named) CONTENT OK

The Base command "Delete Marker" erases one marker from the file you have loaded. Do not include a pound sign (#) in your CONTENT specification (unless you made it part of the marker itself when you gave the Mark command). If you want to delete

all the markers in a file, use the Delete All markers command. To see a list of the markers in your file use the command Show Marker (list). See also: marker, Mark. 4D4E

TNLS example:
BASE C: Delete C: Marker (named) T/[A]: roo
OK:
BASE C: 4D4E1

Modifications: Delete Modifications (to file) OK (really?) OK
The Base command "Delete Modifications" discards all changes you have made on a file since the last update. The file will return to the state it was in after the last update. When you edit a file, the changes build a set of modifications. The Update command fixes the changes permanently in the file. 4D4F

TNLS example:
BASE C: Delete C: Modifications (to file) OK:
(really?) OK:
BASE C: 4D4F1
modification file
##<nls, modification>##

Window: [DISPLAY only]: Delete Window (at) BUG OK
The Base command "Delete Window" erases a window created by the Break Window command. It will keep the window in which the cursor resides when you hit OK. Once a window has been deleted, it cannot be retrieved. See also: window, Expand Window. 4D4G

Edit: [TYPEWRITER only] Edit Statement (at) DESTINATION EDITSTRING OK
The Base command "Edit Statement" (in TNLS only) allows you to move through a statement character by character, changing as you go by means of a set of special editing characters. You must begin at the first character of the statement, working your way through by way of the control-characters listed below under EDITSTRING. (The command resembles QED and TECO.) 4D5

TNLS example:
BASE C: Edit C: Statement (at) A: 035
T: exa<new>mple edit
BASE C: 4D5A

EDITSTRING
any combination of the following control-characters. Any other character you type replaces the character at your current location. At confirmation, the remainder of the old statement copies to the new one; the new one then replaces the old. Control-characters are disabled from their other meanings when using the "Edit" command except for your Telnet escape character. For this reason, Telnet users should change their escape character to <CTRL-Y>. 4D5B

]<CTRL-F> copies one character.
]<CTRL-U> copies through the end of the old statement.
]<CTRL-Z>c copies characters up to and including c (where c = the next occurrence of the typed-in character following <CTRL-Z>).
]<CTRL-O>c copies characters up to but not including c (where c = the next occurrence of the typed-in character following <CTRL-O>).

]<CTRL-S> skips (deletes) one character from the old string.
]<CTRL-G>c skips (deletes) characters up to and including c (where c = the
next occurrence of the typed-in character following <CTRL-G>).
]<CTRL-P>c skips (deletes) characters up to but not including c (where c =
the next occurrence of the typed-in character following <CTRL-P>).
]<CTRL-E>TYPEIN<CTRL-E> enters into the new statement the character string
(TYPEIN) between the two <CTRL-E>s without affecting where you
are in the old statement.
]<CTRL-H> backspaces (deletes) one character in the new without affecting
where you were in the old statement.
]<CTRL-W> backspaces (deletes) one word in the new without affecting where
you were in the old statement.
]<CTRL-Q> voids all editing prior to pressing this key. (The key moves
you back to the beginning of the statement.)
]<CTRL-N> deletes the last character in the new and moves you back one
character in the old statement (a one-character "restorative
backspace).
]<CTRL-R> reprints the existing part of the new statement (but does not
effect editing up to that point).

]See also: editing, ctrl-character (for control characters)
%Nouns.
Statement: ##<edit>## 4D5D1
Branch:
You can only Edit one Statement at a time. Use ##<edit>## 4D5D2
Group:
You can only Edit one Statement at a time. Use ##<edit>## 4D5D3
Plex:
You can only Edit one Statement at a time. Use ##<edit>## 4D5D4

Enlarge (window) SPOT (to) SPOT OK [DISPLAY only]
The Base command "Enlarge (window)" allows you to change the boundaries of windows you have created on your display screen with the Break Window command. For the first SPOT, you BUG the existing boundary; for the second SPOT, BUG the place where you want the boundry to go. See window, Delete Window. 4D6

Force (case)
The Base command "Force" allows you to capitalize, decapitalize, or make first letter capitals in the situations listed below. You can also switch from the default, which capitalizes everything, to a different mode. 4D7
STRING: Force (case) STRING (at) DESTINATION [CASEMODE] OK:

CASEMODE = Upper, Lower, or First (letter upper)
Th Base "Force (CASE) STRING" command changes the case of characters in the chosen STRING according to the case mode currently in effect. The mode has been set by default to "upper" (all characters capitalized); or by the user with the Reset Case (mode) command (to Upper); or Force (Case) Mode command (to Upper or Lower or First letter upper). The most recently set mode determines the result of this command unless you use the [CASEMODE] option to temporarily override it. You can also change the case of letters in a whole statement or branch, etc., with the Force (Case) STRUCTURE commands. 4D7A

CASEMODE
Precede one of the following by the OPTION character <CTRL-U>
##<mode !casemode>## 4D7A1
STRING = Character, Text, Word, Visible, Invisible, Number, or Link
##<nls, string>##
Modify subsystem Force (sentence case in) STRUCTURE command
##<modify, force>##
%Nouns.
Character:
Force case Character is a special case of Force ##<force !string>## 4D7A4A
Word:
Force case Word is a special case of Force ##<force !string>## 4D7A4B
Visible:
Force case Visible is a special case of Force ##<force !string>## 4D7A4C
Invisible:
Force case Invisible is a nonfunctional case of Force ##<force !string>## 4D7A4D
Number:
Force case Number is a nonfunctional case of Force ##<force !string>## 4D7A4E
Link:
Force case Link is a special case of Force ##<force !string>## 4D7A4F
Text:
Force case Text is a special case of Force ##<force !string>## 4D7A4G

STRUCTURE: Force (case) STRUCTURE (at) DESTINATION [CASEMODE]
OK:
CASEMODE = Upper, Lower, or First (letter upper)
The Base "Force (CASE) STRUCTURE" commands change the case of characters in the chosen STRUCTURE according to the case mode currently in effect. The mode has been set by default to "Upper" (all characters capitalized); or by the user with the Reset Case Mode command (to Upper); or Force (Case) Mode command (to Upper, or Lower, or First letter upper). The most recently set mode determines the result of this command unless you use the [CASEMODE] option to temporarily override it. You can also set the case of parts of statements with the Force (Case) STRING commands. 4D7B

CASEMODE
Precede one of the following by the OPTION character <CTRL-U>
##<mode !casemode>## 4D7B1
STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.
Statement:
Force case Statement is a special case of Force ##<force !structure>## 4D7B3A
Branch:
Force case Branch is a special case of Force ##<force !structure>## 4D7B3B
Plex:
Force case Plex is a special case of Force ##<force !structure>## 4D7B3C
Group:
Force case Group is a special case of Force ##<force !structure>## 4D7B3D

Mode: Force (case) Mode CASEMODE OK:
CASEMODE = Upper or Lower or First (letter upper)
The Base command "Force (Case) Mode" allows you to choose whether subsequent Force STRING and Force STRUCTURE commands make all letters lower case, capitalize all letters, or capitalize only the first letter of every word. The Force case STRING and Force case STRUCTURE commands will work in the CASEMODE you choose here (unless they're overridden with the optional temporary CASEMODE in that command) until this command or Reset Case (mode) is used again. 4D7C
CASEMODE: Upper, Lower, or First (letter upper)
a variable in the Base commands for adjusting case mode. Upper sets all alphabetic characters to upper case. Lower sets all alphabetic characters to lower case. First (letter upper) sets the first alphabetic character only of all words to uppercase.
[Side-effect: If you Force Text in First (letter upper) and the text starts in the middle of a word, the first character of the text (not the word it begins in) will be capitalized.] See also: Force (Case), Reset Case. 4D7C1

Insert
##<insert>##
%Merge: Merge STRUCTURE (at) DESTINATION (into) DESTINATION OK
This command is not implemented. This command moves and combines a plex, a group, or the substructure of a branch into another of the same kind. "Merge Statement" has no meaning. If both STRUCTUREs have been previously sorted with the Sort command, the merged STRUCTURE will be properly sorted. After execution, you are at the first character of the first statement of the new group or plex.
TNLS example:
BASE C: Merge C: Plex (at) A: 2a1
(into) A: 5b1
BASE C: 4D9A

Move
The Base command "Move" transfers one of the following SOURCES you specify to the DESTINATION you specify. It combines the Copy

and Delete commands. 4D10
STRING: Move STRING (from) SOURCE (to follow) DESTINATION OK
The Base command "Move STRING" moves a STRING to another location. SOURCE includes the choice of TYPEIN, that is, you can "insert" a new STRING, instead of "moving" one, into the new location with this command. After execution, you are at the last character of the STRING in its new place. 4D10A
TNLS example:
BASE C: Move C: Character (from) A: 3d "use."
(to follow) A: "re)"
BASE C: 4D10A1
STRING = Character, Text, Word, Visible, Invisible, Number, or Link
##<nls, string>##
%Nouns.
Character
The Base command "Move Character" is a special case of Move ##<move !string>## 4D10A3A
Word
The Base command "Move Character Word is a special case of Move ##<move !string>## 4D10A3B
Visible
The Base command "Move Visible" is a special case of Move ##<move !string>## 4D10A3C
Invisible
The Base command "Move Invisible" is a special case of Move ##<move !string>## 4D10A3D
Number
The Base command "Move Number" is a special case of Move ##<move !string>## 4D10A3F
Link
The Base command "Move Link" is a special case of Move ##<move !string>## 4D10A3F
Text
The Base command "Move Text" is a special case of Move ##<move !string>## 4D10A3G
STRUCTURE: Move STRUCTURE (from) SOURCE (to follow) DESTINATION [(Filtered:) FILTER] LEVEL-ADJUST OK
The Base "Move STRUCTURE" command moves a STRUCTURE to another location. After executing the Move command, you are at the first character of the STRUCTURE in its new place. Note--Move does not pay attention to the viewspecs you have on (i.e., it works throughout the STRUCTURE you point to whether all of it is visible to you or not), unless you specify VIEWSPECS in the FILTER option, which pays attention to level, content analyzer, names, and sequence generator viewspecs. You cannot limit the effects to certain lines (like first lines only) with a filter. 4D10B
TNLS example:
BASE C: Move C: Group (from) A: 3d1
(through) A: 3d5
(to follow) A: 2a
L: d
BASE C: 4D10B1
FILTER option: enter VIEWSPECS
With this option in force the only statements that will

move are those that "pass" certain viewspecs you can specify as a string of character codes. Type the OPTION character <CTRL-U> first and then enter any combination of the following viewspecs: for level--a,b,c,d,e,w,x; for content analyzer--i,j,k; for sequence generator--O,P. If the filtering leaves statements somewhere without a source, substatements will move up in level. After using this option, you will be able to specify a LEVEL-ADJUST. See also: substatement, level.
STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.
Statement
The Base command "Move Statement" is a special case of
Move ##<move !structure>## 4D10B4A
Branch
The Base command "Move Branch" is a special case of
Move ##<move !structure>## 4D10B4B
Plex
The Base command "Move Plex" is a special case of Move
##<move !structure>## 4D10B4C
Group
The Base command "Move Group" is a special case of
Move ##<move !structure>## 4D10B4D
Edge: This command has been changed to Enlarge (window).
##<enlarge>## 4D10C
File: Move File (from old filename) CONTENT (to new filename) CONTENT OK
The Base command "Move File" transfers a file and its partial copy from one directory to another, changing the name of the file to the name you give. It is the equivalent of the Executive System Rename file command. You cannot move a file that is loaded. The old name will appear in the origin statement until you Update the file. See effects, below, for more information about typing in CONTENT. 4D10D
TNLS example:
BASE C: Move C: File (from old filename) T: wizard
(to new filename) T: user, wizard
Moved Files Are:
< GIZ, WIZARD.NLS;1, > [and PC] to < USER, WIZARD.NLS;1, >
[and PC]
BASE C: 4D10D1
effects
Move File will produce slightly different results depending on what you type in for CONTENT: 4D10D2
1) When a version number is not specified for the first file name, all versions of the file are moved, and the new names are given the same version numbers as the old names. 2) If you want to specify a version number in your new file name, you must also specify a version number when you are typing in the name of the file being moved. Only one version of the file is then moved and renamed. 3) You cannot move a file from one directory to another if there is already a file with the same name and number in the second directory, or if you rename your file to a name and number already existing in the

second directory.

Replace

The Base command "Replace" allows you to erase one of the following things at the DESTINATION you specify and put in some other CONTENT you specify. It combines the Insert and Delete commands into one command. 4D11

STRING: Replace STRING (at) DESTINATION (by) CONTENT OK

The Base command "Replace STRING" erases an existing STRING and replaces it with another of the same kind that you specify for CONTENT. You have the choice of pointing to a STRING to be copied as a replacement, or of typing in one from the keyboard. After execution you are at the last character of the new STRING. When replacing a link, angle-bracket delimiters will be automatically supplied if you supply no delimiters. 4D11A

Warning when replacing links

##<nls, link !warning>##

TNLS example:

BASE C: Replace C: Character (at) A: 7a ">"

by T: "."

BASE C: 4D11A2

special effects when using Replace Number

##<nls, replacing>##

STRING = Character, Text, Word, Visible, Invisible, Number, or Link

##<nls, string>##

%Nouns.

Character

The Base command "Replace Character" is a special case of Replace ##<replace !string>## 4D11A5A

Word

The Base command "Replace Word" is a special case of Replace ##<replace !string>## 4D11A5B

Visible

The Base command "Replace Visible" is a special case of Replace ##<replace !string>## 4D11A5C

Invisible

The Base command "Replace Invisible" is a special case of Replace ##<replace !string>## 4D11A5D

Number

The Base command "Replace Number" is a special case of Replace ##<replace !string>## 4D11A5E

Link

The Base command "Replace Link" is a special case of Replace ##<replace !string>## 4D11A5F

Text

The Base command "Replace Text" is a special case of Replace ##<replace !string>## 4D11A5G

STRUCTURE: Replace STRUCTURE (at) DESTINATION (by) CONTENT OK

The Base command "Replace STRUCTURE" erases an existing STRUCTURE and replaces it with another of the same kind that you copy, or with something that you type in. For CONTENT you may point to a STRUCTURE to be copied as a replacement, or type in a statement. After execution you are at the first character of the new STRUCTURE. 4D11B

TNLS example:

(To replace a plex with a single statement:)
BASE C: Replace C: Plex (at) A: 2a
(by) T: It was all lies.
BASE C: 4D11B1
STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.
Statement
The Base command "Replace Statement" is a special case of Replace ##<replace !structure>## 4D11B3A
Branch
The Base command "Replace Branch" is a special case of Replace ##<replace !structure>## 4D11B3B
Plex
The Base command "Replace Plex" is a special case of Replace ##<replace !structure>## 4D11B3C
Group
The Base command "Replace Group" is a special case of Replace ##<replace !structure>## 4D11B3D

Reset Case (mode)
##<reset !case>##
Sort STRUCTURE (at) DESTINATION OK
The Base command "Sort" orders the highest-level statements in a plex (Sort Plex) or group (Sort Group) according to your current sort-key. Statements carry their substructure. "Sort Statement" has no meaning and specifying "Branch" for this command defines a plex (one level down) to be sorted. After execution, you are at the first character of the new group or plex. 4D13
TNLS example:
BASE C: Sort C: Plex A: 4b1
BASE C: 4D13A
sort-keys and effects
##<programs, sort-keys>##
Substitute STRING (in) [(filtered:) VIEWSPECS] STRUCTURE (at) DESTINATION (the new STRING) CONTENT (for all occurrences of the old STRING) CONTENT (Finished?) S/Y/N OK
The Base command "Substitute" allows you to put a new STRING in the place of an old STRING everywhere it appears in the STRUCTURE you specify. Substitute is easier than Replace when you want to make the same replacement in more than one location. Type in the new and old text for CONTENT. If you're in TNLS and want to substitute nothing for the old STRING, just give a Command Accept for CONTENT. In DNLS use <CTRL-N>. This is the most common editing command in TNLS. 4D14
S/Y/N
S/Y/N gives you the choice of seeing your current substitution list status (type "s") or cycling through the command again (you are back at "new") and making another substitution in that STRUCTURE. You are limited to 30 substitutions with a maximum of 600 total characters typed in by you.
TNLS example:
BASE C: Substitute C: Visible (in) C: Branch (at) A: 0
(New VISIBLE) T: $1000.50
(Old visible) T: $1.50

```
    Finished? Y/N/S: Yes OK:
    Substitute In Progress
    Substitutions made: 8
    BASE C:                                                    4D14B
     FILTER option
    ##<nls, filter>##
     STRING = Character, Word, Text, Visible, Invisible, Number,
    Link
    ##<nls, string>##
     STRUCTURE = Statement, Branch, Group, Plex
    ##<nls, structure>##
  Transpose
    The Base command "Transpose" allows you to make STRINGS of the
  same kind or STRUCTURES of the same kind interchange places.  4D15
    STRING:  Transpose STRING (at) DESTINATION (and) DESTINATION
    OK
      The Base command "Transpose STRING" makes two STRINGS of the
    same kind replace each other.  After execution, you are at the
    first character of the STRING you selected first.          4D15A
      TNLS example:
      BASE C: Transpose C: Word (at) A: 1 "herring"
      (and) A: "red"
      OK:
      BASE C:                                                  4D15A1
       STRING = Character, Text, Word, Visible, Invisible,
      Number, or Link
      ##<nls, string>##
      %Nouns.
        Character
          The Base command "Transpose Character" is a special
        case of Transpose ##<transpose !string>##            4D15A3A
        Word
          The Base command "Transpose Word" is a special case of
        Transpose ##<transpose !string>##                    4D15A3B
        Visible
          The Base command "Transpose Visible" is a special case
        of Transpose ##<transpose !string>##                 4D15A3C
        Invisible
          The Base command "Transpose Invisible" is a special
        case of Transpose ##<transpose !string>##            4D15A3D
        Number
          The Base command "Transpose Number" is a special case
        of Transpose ##<transpose !string>##                 4D15A3E
        Link
          The Base command "Transpose Link" is a special case of
        Transpose ##<transpose !string>##                    4D15A3F
        Text
          The Base command "Transpose Text" is a special case of
        Transpose ##<transpose !string>##                    4D15A3G
    STRUCTURE:  Transpose STRUCTURE (at) DESTINATION (and)
    DESTINATION [(Filtered:) FILTER] OK
      The Base command "Transpose STRUCTURE" makes two STRUCTURES
    of the same kind replace each other.  After execution, you are
    at the first character of the STRUCTURE you selected first.
                                                             4D15B
      TNLS example:
```

BASE C: Transpose C: Statement (at) A: 1a
(and) A: 1
OK:
BASE C:
4D15B1

FILTER option
This option transposes only statements that "pass" certain viewspecs you can specify as a string of character codes. Type the OPTION character <CTRL-U> first and then enter any combination of the following viewspecs: for level--a,b,c,d,e,w,x; for content analyzer--i,j,k; for sequence generator--O,P. If the filtering leaves statements somewhere without a source, substatements will move up in level.

STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.

Statement
The Base command "Transpose Statement" is a special case of Transpose ##<transpose !structure>##
4D15B4A

Branch
The Base command "Transpose Branch" is a special case of Transpose ##<transpose !structure>##
4D15B4B

Plex
The Base command "Transpose Plex" is a special case of Transpose ##<transpose !structure>##
4D15B4C

Group
The Base command "Transpose Group" is a special case of Transpose ##<transpose !structure>##
4D15B4D

Update...
The Base command "Update" incorporates your modifications into your file and updates the status information in the origin statement. See origin modifying.
4D16

Compact: Update (file) Compact OK
The Base command "Update (file) Compact" updates your file in such a way as to use the file space efficiently. This saves on disk pages, but takes more computer time to execute. Note: To help prevent bad files, it is a good practice to use this command periodically on large files. See also: modification file, version.
4D16A

TNLS example:
BASE C: Update (file) C: Compact OK:
BASE C:
4D16A1

New: Update (file) New OK
The Base command "Update New" permanently incorporates the changes you have made in a file and creates a new version of the file that includes the changes you have made. The old version will look as the file did after the previous update. At any time, before you use the Update New command, you may discard all changes with the Delete Modifications command.
4D16B

modification file:
##<nls, modification>##

Old: Update (file) Old (version) OK
The Base command "Update (file) Old" incorporates the changes you have made into the old version. This is a bit dangerous; if any problems occur during the Update, you may

not have a prior version to back up to. It is useful if you are short on disk space. See also: modification file, version. 4D16C

Rename: Update (file) Rename (to filename) CONTENT OK

The Base command "Update (file) Rename" allows you to Update to a file with a new name instead of a new version of the current file. You must specify a name for the new file. See also: modification file, version. 4D16D

effects

You may duplicate a file in another directory by giving it a name that begins with the name of the second directory. This will only work if you have the right to write on that directory. See also: accessing. 4D16D1

Delete Modifications

##<delete !modifications>##

%Undelete Modifications

##<undelete !modifications>##

<ENTER> (<CTRL-E>)

##<nls, okinsert>##

viewing-commands: Load File, Output, Print, etc.

The Base commands in this group allow you to read and view files, move around to different locations in files, and print files at your terminal and on a line printer. See also: reading, viewing, moving, Show. 4E

Linefeed (<CTRL-J>)

Typing the <LINEFEED> key in TNLS will print the next statement after the statement where you are. On most terminals, the character for this function is the <LINEFEED> key or <CTRL-J>. No <OK> is required in this command. After execution, you are at the first character of the statement printed. To do the equivalent of the <LINEFEED> key in DNLS, use the Jump (to) Next command. See also: next. 4E1

TNLS example:

BASE C: <LF>

3b Of shoes and ships and ceiling wax, of cabbages and kings.

BASE C: 4E1A

Load File CONTENT OK

The Base command "Load File" will call up an already existing file that you specify with a FILEADDRESS for CONTENT. You will be moved to the first character of the origin statement in the file. 4E2

TNLS example:

BASE C: Load C: File T: USERGUIDES,LOCATOR,

< USERGUIDES, LOCATOR.NLS;10 >

BASE C: 4E2A

Output

In general, Output means copying an NLS file to some printed form or to a computer file in a sequential form. The following Output commands are available in the Base subsystem. For information about the tools to format files for printing, See publication. See also: sequential file, Print. 4E3

% Branch 2b5c, containing the Output commands, has been cleaned up as of 9/9/75, with the exception of the Output Remote command, which needed more clarification. However, it does not reflect the proposed changes.

Assembler: 4E3B
File: Output (to) Assembler File CONTENT OK
The Base command "Output Assembler File" allows you to convert an NLS file into a sequential file with each statement ending with a carriage return and linefeed. Specify a FILEADDRESS for CONTENT. The filename extension will default to "TXT". See also: Output Sequential. 4E3B1

TNLS example:
BASE C: Output (to) C: Assember C: File T: prog
OK:
Output for Assembler in Progress
BASE C:
Force: Output (to) Assembler File CONTENT Force (upper case) OK
The Base command "Output Assembler File Force"" does the same thing as Output Assembler File and, in addition, makes the sequential file upper case. 4E3B1B

Append: Output (to) Assembler Append (to file) CONTENT OK
The Base command "Output Assembler Append" allows you to convert an NLS file into a sequential file with each statement ending with a carriage return and linefeed, and append that file to an existing sequential file that you specify for CONTENT. See also: Output Sequential Append. 4E3B2

Force: Output (to) Assembler Append (to File) CONTENT Force (upper case) OK
The Base command "Output Assembler Append Force" does the same thing as Output Assembler Append and, in addition, makes the sequential file to be appended upper case. 4E3B2A

COM: Output (to) Com (for device) DEVICE OK
The Base command "Output (to) Com" allows you to produce from the file you have loaded a sequential file formatted for Computer Output to Microfilm. You must specify one of three phototypesetting devices: Comp80, Singer, or Videocomp. The command creates a job identified by your initials in the directory COM. Output Processor directives are considered in formatting the file. Printing begins from the statement where you are located. After your compiled file is in the COM directory, contact Feedback. See publication com. See also: directives, Output Printer. 4E3C

DEVICE = Comp80
Singer
Videocomp 4E3C1

TNLS example:
BASE C: Output C: Com (for device) Singer OK:
Processing Output
BASE C: 4E3C2

Append: Output (to) Com (for device) DEVICE Append (to file) CONTENT OK
The Base command "Output (to) Com (for device) DEVICE Append" produces a sequentail file formatted for Computer Output to Microfilm at the FILEADDRESS you specify for CONTENT. If no directory is specified, it will appear in your directory. 4E3C3

File: Output (to) Com (for device) DEVICE File CONTENT OK
The Base command "Output (to) Com (for device) DEVICE File" produces a sequential file formatted for Computer Output to Microfilm at the FILEADDRESS you specify for CONTENT. If no directory is specified, it will appear in your directory. 4E3C4

Copies: Output (to) Com (for device) DEVICE Copies CONTENT OK
This command probably does not have any useful function. The Base command "Output (to) Com (for device) DEVICE Copies" produces a sequential file formatted for Computer Output to Microfilm and creates the number of copies of that file that you specify for CONTENT in the COM directory. No file will be created in your directory. 4E3C5

Test: Output (to) Com (for device) DEVICE Test OK
The Base command "Output (to) Com (for device) DEVICE Test" formats the file, places special symbols in it to represent what the COM version would look like when printed, and prints the test on the line printer. 4E3C6

File: Output (to) Com (for device) DEVICE Test File CONTENT OK:
The Base command "Output (to) Com (for device) DEVICE Test File" creates a sequential file containing the COM test. By using Input Sequential you can then create an NLS file from your sequential file, and view this file at your terminal. This is particularly helpful for content searches in COM tests. 4E3C6A

Output Processor--the program that formats files for printing
##<output !processor>##
for a dicussion of publication through COM from NLS
##<publication, photocomposition>##

Journal: Output (to) Journal (quickprint) OK
The Base command "Output (to) Journal" searches the file you have loaded for a branch named "journal", such as in your initial file. On your computer's line printer it then prints one copy of each journal citation followed by the complete item (except for the origin statement of a file). To print only those items you have not already seen, you must move the items you have seen out of the branch named "journal". See also: journal, statementname, Move STRUCTURE, Sendmail, initial. 4E3D

TNLS example:
BASE C: Output (to) C: Journal (quickprint) OK:
Output Journal Quickprint in Progress
BASE C: 4E3D1

Append: Output (to) Journal (quickprint) Append (to file) CONTENT OK
The Base command "Output (to) Journal (quickprint) Append" appends the sequential file generated from journal items to the sequential file you specify for CONTENT. See also: Sendmail. 4E3D2

Copies: Output (to) Journal (quickprint) Copies CONTENT OK
The Base command "Output (to) Journal (quickprint) Copies" prints on the line printer as many copies as you specify for CONTENT. This command prints from the file you

have loaded anything you have in a branch named "journal." Each journal item and the complete text of any links that are shown are printed. See also: Sendmail. 4E3D3

File: Output (to) Journal (quickprint) File CONTENT OK
The Base command "Output (to) Journal (quickprint) File" produces a sequential file named whatever you specify for CONTENT from the citations and complete text of links in the branch called "journal" from your loaded file. See also: Sendmail. 4E3D4

No: Output (to) Journal (quickprint) No (leaders) OK/C
The Base command "Output (to) Journal (quickprint) No (Headers)" prints your journal mail so that the only thing appearing at the top-right of each page is "Page #". See also: Output Quickprint. Instead of OK, you may also specify these command words: 4E3D5

Append: Output (to) Journal (quickprint) No (headers) Append (to file) CONTENT OK
The Base command "Output (to) Journal (quickprint) No (Headers) Append" allows you to specify that your quickprint file be appended to a sequential file. When prompted for CONTENT, give the FILEADDRESS of a sequential fle. 4E3D5A

Copies: Output (to) Journal (quickprint) No (headers) Copies CONTENT OK
The Base command "Output Journal (quickprint) No (Headers) Copies" does the same thing as "Output Journal (quickprint) No" and allows you to specify for CONTENT the number of copies to be printed. 4E3D5B

File: Output (to) Journal (quickprint) No (headers) File CONTENT OK
The Base command "Output (to) Journal (quickprint) No (Headers) File" does the same thing as "Output Journal (quickprint) File", except that the sequential file has no headers. 4E3D5C

% backlinks: <sendmail, journal>

Printer: Output (to) Printer OK
The Base command "Output (to) Printer" will produce a sequential file in the ARC computer's printer directory formatted for the line printer. Output Processor directives are considered in formatting the file. Printing begins from the statement where you are located. Any files in the printer directory will be printed, then deleted. If you wish to save the print file, use Output (to) Printer File, below, to specify a filename in another directory and copy it to the printer directory when you want to print it. See <userguides,op-guide,3>. See also: directives, Outprint Quickprint, Output Com. 4E3E

TNLS example:
BASE C: Output (to) C: Printer OK:
Processing Output
BASE C: 4E3E1

Copies: Output (to) Printer Copies CONTENT OK
The Base command "Output (to) Printer Copies" does the same thing as Output Printer, and also allows you to print as many copies on the line printer as you specify for CONTENT. 4E3E2

File: Output (to) Printer File CONTENT OK
The Base command "Output (to) Printer File" creates a sequential file formatted for the line printer named whatever you specify for CONTENT. Use this command when you do not wish to print a file immediately, then copy it to your computer's printer directory when you are ready to print. See Output Printer. 4E3E3

Append: Output (to) Printer Append (to file) CONTENT OK
The Base command "Output (to) Printer Append" creates a sequential file formatted for the line printer and then appends this file to the sequential file you specify for CONTENT. See Output Printer. 4E3E4

Output Processor--the program that formats files for printing:
##<output !processor>##
% link to op userguide.

Quickprint: Output (to) Quickprint OK
The Base command "Output (to) Quickprint" produces a sequential file formatted for a line printer. It observes viewspecs in force but not Output Processor directives, which remain as text in the printout. The printout is paginated and bears the file header. Quickprint runs more than 10 times faster than Output Printer. Output Quickprint compiles your file to a printer file with the same view you would get with a Print command, but paginated. Printing begins from your current location. When you follow the command with <OK> rather than specifying any of the following alternatives, the file will go into the <ARCPRINTER> directory and be printed at ARC. See also: directives. 4E3F

TNLS example:
BASE C: Output (to) C: Quickprint OK:
Output (to) Quickprint in Progress
BASE C: 4E3F1

Append: Output (to) Quickprint Append (to file) CONTENT OK
The Base command "Output (to) Quickprint Append" allows you to specify for CONTENT the FILEADDRESS of a sequential file to which your quickprint file is to be appended. 4E3F2

Copies: Output (to) Quickprint Copies CONTENT OK
The Base command "Output (to) Quickprint Copies" allows you to specify for CONTENT the number of copies to be quickprinted at the line printer where you are logged in. 4E3F3

File: Output (to) Quickprint File CONTENT OK
The Base command "Output (to) Quickprint File" creates a sequential file named whatever you specify for CONTENT. Use this command whenever you do not wish to quickprint a file immediately, then copy it to your computer's printer directory when you are ready to print it. Any files in the printer directory will be printed and then deleted. See Output Quickprint. See also: Copy File. 4E3F4

No: Output (to) Quickprint No (headers) OK/C
The Base command "Output (to) Quickprint No" prints your file so that the only thing appearing at the top-right of each page is "Page #". See also: Output Quickprint. Instead of OK, you may also specify these commandwords: 4E3F5

Append: Output (to) Quickprint No (headers) Append (to file) CONTENT OK
The Base command "Output (to) Quickprint No (Headers) Append" allows you to specify for CONTENT the FILEADDRESS of a sequential file to which your quickprint is to be appended. 4E3F5A
Copies: Output (to) Quickprint No (headers) Copies CONTENT OK
The Base command "Output Printer No (Headers) Copies" does the same thing as "Output Quickprint No" and allows you to specify for CONTENT the number of copies to be printed. 4E3F5B
File: Output (to) Quickprint No (headers) File CONTENT OK
The Base command "Output (to) Quickprint No (Headers) File" does the same thing as "Output Quickprint File", except that the sequential file has no headers. 4E3F5C

%Test:
I think this command must do the same as ##<output !quickprint >##

Remote: Output (to) Remote (printer -- TIP) CONTENT (Port #) CONTENT OK...
The Base command "Output Remote" will create a print file, considering directives, and send it to a printer on a network TIP. To use Output Remote you must have a printer connected to a dedicated tip port at your site, and you must supply the number of the available TIP port to the command. You may choose not to send form feeds (page breaks); they may be simulated with line feeds, or you can send no page breaks at all. You may have the printer wait after each page if you like. You may also have it wait to start until you say go. 4E3G

TNLS example:
BASE C: Output (to) C: Remote (printer--TIP) T: radc-tip (Port #) T: 4
(Send Form Feeds?) Y/N:
(Wait at page breaks?) Y/N:
(CONFIRM when ready) OK:
Output Remote printer in Progress
BASE C: 4E3G1
syntax:
Output (to) Remote (printer -- TIP) CONTENT (port #) CONTENT OK (Send Form Feeds?) CHOICE (Wait at page break?) ANSWER (CONFIRM when ready) OK
CHOICE = Yes or
No (simulate?) ANSWER
ANSWER = Yes or OK (same as Yes) or
No
form feed
##<nls, ctrl-l>##
line feed
##<nls, lf>##

Sequential
See also: Copy Sequential. 4E3H
File: Output (to) Sequential File CONTENT OK
The Base command "Output Sequential File" converts an NLS

file into a sequential file with each line ending with a carriage return and linefeed. Specify a FILEADDRESS for CONTENT. The filename extension will default to "TXT". See also: Output Assembler. 4E3H1

TNLS example:
BASE C: Output (to) C: Sequential C: File T: buz
OK:
Output Sequential in Progress
BASE C: 4E3H1A

Append: Output (to) Sequential Append (to file) CONTENT OK
The Base command "Output Sequential Append" appends the sequential output from your file to the sequential file whose FILEADDRESS you specify for CONTENT. An NLS file may be converted to a sequential text file with the Output Sequential File command. The filename extension will default to "TXT". See also: Output Assembler Append. 4E3H2

sequential files (Executive System):
<core, sequential>

Terminal: Output (to) Terminal OK...
The Base command "Output (to) Terminal" will process a file considering directives, as does the Output (to) Printer command, but it will print at your terminal. You may choose not to send form feeds when a new page starts; they may be simulated with line feeds. You may have the output wait after each page if you like. 4E3I

TNLS example:
BASE C: Output (to) C: Terminal OK/C: OK:
(Send Form Feeds?) Y/N:
(Wait at page break?) Y/N:
(CONFIRM when ready) OK:
Processing Output

BASE C: 4E3I1

syntax:
Output (to) Terminal OK: (send form feeds?) CHOICE (wait at page break?) ANSWER (CONFIRM when ready) OK
CHOICE = Yes or
No (simulate?) ANSWER
ANSWER = Yes or OK (same as Yes) or
No

File: Output (to) Terminal File CONTENT OK...:
The Base command "Output (to) Terminal File" will create a sequential file, formatted by Output Processor directives, that may then be sent via sndmessage, copied over the ARPANET, or sent to the printer. <CTRL-O> will abort the printing. You may give instructions about having form feeds sent or having output wait at page breaks. See also: Output Terminal, Output Printer, Output Sequential. 4E3I3

syntax:
Output (to) Terminal File OK: (send form feeds?) CHOICE (wait at page break?) ANSWER (CONFIRM when ready) OK
CHOICE = Yes or
No (simulate?) ANSWER
ANSWER = Yes or OK (same as Yes) or
No

] See also:
processor: the Output Processor program
The program that formats an NLS file for printing on a line printer, teletype, microfilm, or other output device. The commands Output Terminal, Output Printer, Output Remote, and Output COM invoke the Output Processor. In making the printed copy the Output Processor formats the file, breaking the text into pages, numbering them, setting up margins, etc. You may request a copy of the Output Processor's Users' Guide from FEEDBACK. See directives. 4E3K
how to use the Output Processor in publication work
##<publication, introduction>##
Print (in TNLS)
The Base (TNLS) command "Print" types at your terminal what you specify. See also: printing. 4E4
File: Print File OK
The Base command "Print" types at your terminal your entire file with current viewspecs without affecting your current location. To select a starting point and view of a file for printing, use the Print STRUCTURE command. See also: Print Rest. 4E4A
effects
Typing <CTRL-O> will stop the printing. After execution, you are at the same statement you were when you started this command. With the default viewspec capital-E turned on, TNLS will paginate your output with any Print command. To turn pagination off, use viewspec capital-F. See also: CTRL-character. 4E4A1
Journal: Print Journal (mail) OK
The Base command "Print Journal" will look for a statement in the file you have loaded named "journal", (as, for example, in your initial file). Then it prints at your terminal each journal citation followed by the complete item. To only get those items you have not already seen, you must move the items you have seen out of the branch named "journal". See also: statementname, Move STRUCTURE, Sendmail, citation, item, Output Journal. 4E4B
Rest: Print Rest OK
The Base (TNLS) command "Print Rest OK" will print the rest of your loaded file, starting with the statement you are currently at. This command will not affect your location or your viewspecs. To select a starting point and view of a file for printing, use the Print STRUCTURE command. Typing <CTRL-O> will stop the printing. See Print STRUCTURE, Print File. 4E4C
STRUCTURE: Print STRUCTURE (at) DESTINATION VIEWSPECS OK
This Base subsystem group of "Print" commands prints at the terminal the particular STRUCTURE in a file you specify, according to the viewspecs you give to control the format of the printout. If you give no viewspecs, your current viewpecs will control printing. 4E4D
TNLS example:
BASE C: Print C: Branch (at) A: 3
V: m
3 (alice) The Walrus
3a "The time has come," the Walrus said,

3b "To speak of many things,
3b1 Of shoes and ships and ceiling wax,
3b2 of cabbages and kings,
3b3 Of why the sea is boiling hot,
3b4 And whether pigs have wings."
BASE C: 4E4D1
effects
Typing <CTRL-O> will stop the printing. After execution, you will be at the first character of the first statement printed, i.e., the statement you were in when you started this command. Any viewspecs you entered remain in effect until you change them again. 4E4D2
STRUCTURE = Statement, Branch, Plex, or Group
##<nls, structure>##
%Nouns.
Statement:
Print Statement is a special case of Print ##<print !structure>##
Branch:
Print Branch is a special case of Print ##<print !structure>##
Plex:
Print Plex is a special case of Print ##<print !structure>##
Group:
Print Group is a special case of Print ##<print !structure>##
% backlinks: <publication, printing>
Jump command:
##<nls, jump>##
Reset Content (pattern)
##<reset !content>##
Reset Viewspecs
##<reset !viewspecs>##
Set Content (pattern)
##<set !content>##
Set Viewspecs
##<set !viewspecs>##
Show Viewspecs
##<base !show !viewspecs>##
TAB-command
Typing the TAB character (usually <CTRL-I>) at the herald in the Base subsystem moves you to the next occurrence (from your present location) of the last character, word, or content search you specified as an ADDRESS element in a content search. No <OK> is required for the execution of this command. If you have not previously used a content search, you will get the message "<tab> valid only to repeat a previous search". For other kinds of tabbing, see tab. See also: Programs content-analyzer, viewspec k. 4E11
TNLS example:
BASE C: <elephant>
BASE C:
BASE C: Print CA:/C:
My elephant eats granola.
BASE C: 4E11A

character search in an ADDRESS
##<nls, infileaddress !characteraddress>##
word or content search in an ADDRESS
##<nls, infileaddress !contentaddress>##
Jump (to) Word command
##<nls, jump !word>##
Jump (to) Content command
##<nls, Jump !content>##

uparrow ^ [TYPEWRITER only]

Typing the charactr ^ (uparrow) at the herald in the Base subsystem will print the statement that is back from (precedes) the statement where you are located. No <OK> is required for the execution of this command. After execution, you are located at the first character of the statement printed. 4E12

TNLS example:
BASE C: ^
3b Of shoes and ships and ceiling wax, of cabbages
and kings,
BASE C: 4E12A

% DNLS: uparrow in DNLS

The DNLS uparrow command (type the character ^ at Base's herald), or a series of uparrow commands, is meant to be used to go back after you have used the <LINEFEED> command(s) to scroll through a file. Each uparrow in the series will return you to the previously scrolled view. No OK is required for execution of this command.

side effects

If a series of uparrows (including a series of one) is used at some time other than immmediately following the same number of <LINEFEED>s, it will work just as inappropriately as Jump to Return does when respecified more than once in succession, e.g, puts the last moved-from location immediately behind the destination, so that the following Jump will take you back to where you just came from, not back one more from the first destination. See also: Jump Return.

]See also: DNLS, herald, Linefeed DNLS, scrolling, OK.

]These 3 commands show where you are:

backslash [TYPEWRITER only]
##<nls, backslash>##

period [TYPEWRITER only]

Typing the period key at the herald of a subsystem prints your statement number, or SID if viewspec capital-I is on, followed by a number indicating how many characters you are to the right of the beginning of the statement. COMPARE address slash. 4E15

TNLS example:
BASE C: . = 2d +12
BASE C: 4E15A

to control periods in printed documents
##<publication, directive !dots>##

slash [TYPEWRITER only]
##<nls, address !slash>##

file-handling commands: Create File, Undelete File, Verify File, etc.

The Base commands in this group allow you to create new files, new versions of old files, and new kinds of files; move files around in

the system; check on the status of files; and renumber SIDs within a file. See also: modifying-commands. 4F

```
 Archive File
##<archive>##
 Copy File
##<copy !file>##
 Copy Sequential File
 ##<copy !sequential>##
Create File CONTENT OK
  The Base command "Create File" makes a new file in a directory.
 It will give the new file the name you specify for CONTENT, load
the file, and insert the origin statement.  You will be at the
first character of the origin statement.  A new file or version
can also be created whenever you update, move, output, or copy a
file.  See also:  directory, origin, accessing, FILEADDRESS.  4F4
   TNLS example:
   BASE C: Create C: File T: birdwatching
   < WREN, BIRDWATCHING.NLS;1, >
   BASE C:                                                    4F4A
    To write in your new file, use Insert Statement.
   ##<insert!statement>##
   ]Other commands that can can create new files or new versions
   of files:
    Update File
   ##<update>##
    Copy File
   ##<copy !file>##
    Output Com File
   ##<output !com !file>##
    Output Journal File
   ##<output !journal !file>##
    Output Printer File
   ##<output !printer !file>##
    Output Quickprint File
   ##<output !quickprint !file>##
 Delete File
##<delete !file>##
 Delete Modifications
##<delete !modifications>##
 Undelete File
##<undelete !file>##
%Undelete Modifications
##<undelete !modifications>##
 Load File
##<load>##
 Move File
##<move !file>##
 Output Assembler File
##<output !assembler !file>##
 Output Sequential File
##<output !sequential !file>##
 Show File Modifications (status)
##<show !file !modifications>##
 Show File Status
##<show !file !status>##
 Set Temporary Modifications
```

##<set !temporary>##
Renumber Sids (in file) OK
The Base command "Renumber Sids" renumbers all the SIDs in the file you're in to be sequential, ignoring the outline structure. SIDs were originally assigned to statements in order of creation.
See also: SID, structure. 4F16
TNLS example:
BASE C: Renumber C: Sids (in file) OK:
BASE C: 4F16A
Reset Archive (request for file)
##<reset !archive>##
Reset Temporary Modifications
##<reset !temporary>##
Undelete
The Base command "Undelete" allows you to bring back one of the following that has been deleted but not expunged. 4F19
File: Undelete File CONTENT OK
The Base command "Undelete File" returns to normal use the deleted file you specify for CONTENT. You may restore the file to normal status at any time before it has been expunged. Once a file has been deleted it will be erased permanently from the system if you use the Expunge command, if the operator runs a program to expunge deleted files, or if you log out. See Expunge Directory, Delete File. See also: Show Directory and look under DIROPT for the Undeleted option. 4F19A
filename
##<nls, fileaddress>##
TNLS example:
BASE C: Undelete C: File T: neargone.NLS;4
Undeleted Files Are
< YOU, NEARGONE.NLS;4, >
BASE C: 4F19A2
Modifications: Undelete Modifications (to file) OK
This command has not been implemented.
The Base command "Undelete Modifications" restores, before an Update, deleted modifications made to a file with the Delete Modifications command. Modifications (made with editing commands, programs...) to a file are kept separate until merged into the file with an Update. All modifications since the last Update can be thrown away with the Delete Modifications command. The command applies only if no modifications have been made to the file subsequent to the Delete Modifications. 4F19B
TNLS example:
BASE C: Undelete C: Modifications (to file) OK: (really?)
OK:
BASE C: 4F19B1
modification files
##<nls, modification>##
Update
##<update>##
Verify File OK
The Base command "Verify File" will check certain aspects of your loaded file. If your file is okay, you will see the message "Successful, internal structure is OK"; if the file is bad, the message "Bad File" will appear. (Sometimes this message appears

when a command cannot be executed because of file difficulties.) The Verify file command does not check every problem with the file, but when it reports a bad file, you have serious problems. 4F21

BAD FILE -- what to do when this message appears
Immediately type: <CTRL-C>reset<CR> (control-C reset carriage-return) and try the following. In the Executive System (TENEX or TOPS-20), type: NLS<CR>. Then load the file and use the Update Compact command. If the Verify File command still says BAD FILE, three courses of action, listed here, can be followed. See also: Executive System. 4F21A

Load an old version
You may have to undelete the old version by using the Undelete File command, or you can get the old version from the nightly dump tape by linking or sending a message to the computer operator. The Deleted option in the Show Directory command lists files deleted but still online (not yet Expunged). If the Verify File command does not say BAD FILE on the old version, then type <CTRL-C>reset<CR> and Delete the BAD versions. The old version will not contain changes you have made since the last time you used the Update command.

version
##<nls, version>##

Create a new file; copy into it the good stucture from the bad file.
Check to make sure all structures you pointed to were copied into the new file. Use the Update Compact command on the file just created. If the Verify File command still says BAD FILE, type <CTRL-C>NLS<CR> and load one of the bad files. Try to determine the location of the bad spot by reading selected portions. Often the problem is in a single statement, usually one you have recently written on. If you can isolate that statement and copy the good parts of the file to a new file, you can save your work. If you use the command Output Terminal, the file should stop printing at the spot that is bad. In DNLS, if the file is so bad that it will not display, you must Simulate TNLS to do this process. In general, use <CTRL-C>NLS<CR> and Update Compact a lot in the process of saving a bad file.
If neither of the above are successful,
you may want to try a third corse of action (not quite as tried and true as the previous two). Turn on viewspec y, and "Output Sequential" your file. Then do a "Copy Sequential Two" on the file to make it an NLS file once again. Some users have had good luck doing this. You do risk turning one or two of your statements into garbage.

TNLS example:
BASE C: Verify C: File OK:
File Verify in Progress
Succesful: internal structure is OK
BASE C:

Or it could look like:

BASE C: Verify C: File OK:

File Verify in Progress
BAD FILE
BASE C: 4F21B

status-commands: Show, Set, Reset

The Base commands in this group allow you to check and adjust the status of things such as viewspecs and name delimiters. 4G

Show

The Base command "Show" allows you to see the status of any one of the following things that you specify. See also: reading, file. 4G1

Directory: Show Directory (of) CONTENT/OK [(opt:)DIROPT] OK

The Base command "Show Directory" will display a list of files in the directory you specify for CONTENT, or in the directory you are connected to if you type <OK>. Escape works when specifying a directory-name. The list of files is shown in link syntax according to options you can specify with commandwords in DIROPT after you type <CTRL-U>. If the file is being modified, it will state this in square brackets following the link syntax for that file. See also: optional, link. 4G1A

TNLS example:
BASE C: Show C: Directory (of) OK/T: T: userguides
OK:
(opt:) C: Size (in Pages) OK: (Finished?) Y/N: OK:
<USERGUIDES...>
...
BASE C: 4G1A1
DIROPT
##<nls, diropt>##
Copy Directory command to insert a directory
##<copy !directory>##
FILEADDRESS and star convention
##<nls, fileaddress !star>##
Connect (to) Directory command
##<connect !directory>##

Disk: Show Disk (space status) OK

The Base command "Show Disk (space status)" will print the number of disk pages in use for deleted and undeleted files, and the maximum number of disk pages allowed for your directory. It also shows how many pages are in use by the whole system, and how many are left. See directory. 4G1B

TNLS example:
BASE C: Show C: Disk (space status) OK:
Connected to WEINBERG
93 Total pages in use -- 300 Allowed, 93 Undeleted, 0 Deleted
System Total: 3246 Pages left, 55604 Used
BASE C: 4G1B1
deleted files
##<delete !file>##

Return: Show Return (ring) OK

The Base command "Show Return (ring)" will show the beginnings of the last few statements where you have been. The most recent statements you visit are added in turn to the top of the list. To change the size of your return ring (from the default 10 to up to 25) use the Useroptions Return (ring

entries) command. See also: statement return. 4G1C

TNLS example:
BASE C: Show C: Return (ring) OK:
so we left for the mountain
and so on
an extraterrestria
buffalo consider.
BASE C: 4G1C1

File: Show File... 4G1D

Default: Show File Default (directory for links) OK
The Base command "Show File Default" will print the default directory name for the file you have loaded (i.e. the name of the directory you would type in for this file in a link). This command has no effect on the file. The default directory for links may be changed with the Set Link command. 4G1D1

TNLS example:
BASE C: Show C: File C: Default (directory for links) OK:
Private File (but with no Access List)
< WEINBERG, RE.NLS;16, >
Default directory for links is WEINBERG
BASE C: 4G1D1A

Modifications: Show File Modifications (status) OK
The Base command "Show File Modifications" will print the modification status of the loaded file. The first creation or change of text begins a set of modifications and changes the modification status to "modified". Normally, only one person can be modifying a file at any time. The file modification status remains "modified" until you incorporate your changes into the file with an Update command. See also: modification file, Update. 4G1D2

TNLS example:
BASE C: Show C: File C: Modifications (status) OK:
< WEINBERG, RE.NLS;17, >
Being Modified By WEINBERG (POOH)
BASE C: 4G1D2A

Return: Show File Return (ring) OK
The Base command "Show File Return" will show your file return ring. The most recent files you've worked in (in this work session) are added to the top of the list; the file you're in will be first. To change the size of your return ring (from the default 10 to up to 25) use Useroptions Filereturn (ring entries) command. See also: Jump File return. 4G1D3

TNLS example:
BASE C: Show C: File C: Return (ring) OK:
< WEINBERG, RE.NLS;17, > Being Modified By WEINBERG (POOH)
< WEINBERG, POOH.NLS;61, > Being Modified By WEINBERG (POOH)
BASE C: 4G1D3A

Size: Show File Size OK
The Base command "Show File Size" will print information about the size of the file in which you are located. It refers to Executive System pages which are roughly

equivalent to typed pages. 4G1D4
TNLS example:
BASE C: Show C: File C: Size OK:
<DOE, WONDER.NLS;34, >
31 statements in file
Structure pages = 1/95
Data pages = 1/370
Total pages in file = 8
1402 words used out of 2048 words in file (=68%)
BASE C: 4G1D4A
Update Compact for efficient use of file space:
##<compact>##
Status: Show File Status OK
The Base command "Show File Status" will print certain basic information about the loaded file, including: default directory for links, modification status, date of creation, and file size. This command has no effect on the file. See also: modification file. 4G1D5
TNLS example:
BASE C: Show C: File C: Status OK:
< WEINBERG, RE.NLS;16, >
Being Modified By WEINBERG (POOH)
Private File (but with no Access List)
Default directory for links is WEINBERG
Creation date of version 1: 14-JAN-75 08:03
Creation date of this version: 6-FEB-75 15:14
26 statements in file
Structure pages = 1/95
Data pages = 6/370
Total pages in file = 8
1101 words used out of 4096 words in file (=27%)
Try an Update Compact to improve % used. 4G1D5A
Update Compact for efficient use of file space
##<compact>##

Marker: Show Marker (list) OK
The Base command "Show Marker" will list the markers, with their addresses, of the loaded file. Markers are normally invisible when viewing your file. They are named with the Mark Character command, and deleted with the Delete Marker or Delete All markers commands. 4G1E

Name: Show Name (delimiters for statement at) DESTINATION OK
The Base command "Show Name" will print the characters currently defined (by the Set and Reset Name delimiters commands or by your Useroptions default) to mark off statementnames for the statement you specify. See also: statementname, name-delimiters. 4G1F
TNLS example:
BASE C: Show C: Name (delimiters for statement at) A: 1
NULL NULL
BASE C: 4G1F1

Viewspecs: Show Viewspecs (status) [Verbose] OK
The Base command "Show Viewspecs" will (in the terse form) list the viewspecs in force in the current work session. Choosing the OPTION Verbose will list the viewspecs with their meanings. The Set and Reset Viewspecs commands are used to change viewspecs. 4G1G

TNLS example:
BASE C: Show C: Viewspecs (status) OK/[**]:
levels: ALL, lines: ALL, hjnpuzACEGJLP
BASE C: 4G1G1
To Show Viewspecs in verbose form...
type the OPTION character <CTRL-U> and then specify the commandword Verbose before confirming the command.
ways of changing viewspecs:
##<nls, viewspecs !changing>##

Set ...
The Base command "Set" allows you to change any one of the following things that you specify. Use the Reset command to get back to the original default setting. See also: Programs Set, Programs Reset. 4G2

Archive: Use the Base Archive command
##<archive>## 4G2A

Character: Set Character (size for window to) CONTENT OK
The Base command "Set Character" allows you to change the size of your characters to one of the numbers (shown below) you specify for CONTENT. Character size 1 is the default. Only the window to which your mouse is pointing when you type <CA> will be affected. This command will not work for terminals that cannot change their character size. See also: Reset Character. 4G2B

]0 The smallest.
]1 The average size (72 characters per line) the default.
]2 Large size.
]3 The largest size.

Content: Set Content (pattern) Off/On/To ...: 4G2C

Off: Set Content (pattern) Off OK
The Base command "Set Content (pattern) Off" de-activates your content-analysis pattern by turning viewspec j on. All statements will pass. Viewspec i or the command "Set Content (pattern) On" activates your content-analysis pattern. 4G2C1

content-analyzer
##<programs, content-analyzer>##

On: Set Content (pattern) On OK
The Base command "Set Content (pattern) On" activates your current content-analysis pattern by turning viewspec i on. Only statements that contain the pattern will pass. Viewspec j or the Base subsystem command "Set Content (pattern) Off" de-activates your content-analysis pattern. 4G2C2

content-analyzer
##< programs, content-analyzer>##
Set Content (pattern) To ...
##<set !content !to>##
Reset Content (pattern)
##<reset !content>##

To: Set Content (pattern) To CONTENT OK
The Base command "Set Content (pattern) To" allows you to type in or point to a content-analyzer pattern. It then compiles the pattern and makes it your current content-analyzer. After you specify the content-analyzer pattern, you may turn it off and on by the viewspecs i, j,

and k. When pointing to a pattern instead of typing it in, you must point to the first character of the entire pattern, which must end with semicolon. Content-analyzer programs already compiled into a program file can be loaded with the Programs subsystem command "Load Program". 4G2C3

content-analyzer
##<programs, content-analyzer>##
content-analysis viewspecs
##<nls, viewspecs !content-analysis>##
square-brackets in content-analyzer patterns
##<programs, square-brackets>##
Programs subsystem Load Program command
##<programs, load>##

Executive: Set Executive (protection for file named) CONTENT CONTROLS OK

The Base command "Set Executive (protection...)" controls access to the file specified for CONTENT. See menu item 1 below for key to syntax. See also: privacy. 4G2D

CONTENT wants a FILEADDRESS.
CONTROLS = Reset
or Allow WHOM DOING (Finished?) ANSWER
or Forbid WHOM DOING (Finished?) ANSWER
or Private (for) WHOM
or Set (to) SIX-DIGITS
WHOM = Self or Group or Public
DOING = Read (access)
or Write (access)
or Execute (access)
or List (access)
or All (access)
or Set (to) TWO-DIGITS
SIX-DIGITS = a TYPEIN of a six octal digit Executive System protection number
TWO-DIGITS = a TYPEIN of a two octal digit Executive System protection number

Executive System protection digits
##<nls, filename !protection>##

effects

In the syntax, "n" for ANSWER cycles you back to DOING. Normally, a file may be read (loaded) by anyone, and changed only by those with write-access to the directory (your group). You may set the protection status of a file to allow or disallow writing and/or reading for you, your group, and others. This command is in addition to the Set Nls (protection ...) command. 4G2D2

write access control
##<write>##

External: Set External (names link file to:) CONTENT OK

To use the Base command "Set External", you must have a file that contains statement names followed by links. Specify this file's name for CONTENT. See also: Jump Name, externalname. 4G2E

Link: Set Link (default for file to directory) CONTENT OK

The Base command "Set Link" allows you change the operation of links in a specific file so they work as if the named directory were in the link. Then, when you do not specify a

directory name for links in that file, the one you have chosen will be assumed. You can go back to the default set by the system prior to this command--the directory the file itself resides in--by using the Reset Link default command. 4G2F

TNLS example:
BASE C: Set C: Link (default for file to directory) T:
alice
BASE C: 4G2F1
defaults for links:
##<nls, fileaddress !defaults>## 4G2F2
Reset Link (default) command
##<reset !link>##
Show File Default (directory for link) command
##<show !file !default>##

Name: Set Name (delimiters in) STRUCTURE (at) DESTINATION (left delimiter) CONTENT (right delimiter) CONTENT OK

The Base command "Set Name" allows you to change the characters that define and mark the boundaries of statement names in a particular STRUCTURE at the DESTINATION you specify. Type in or point to one character as the CONTENT for both the left and right delimiters. It will only take effect on those statements passing the current viewspecs. Any visible that is not a reserved for statement names can be used as a delimiter. See name-delimiters. See also: statementname, Reset Name, Show Name. 4G2G

TNLS example:
BASE C: Set C: Name (delimiters in) C: Branch (at) A: 2a
left delimiter T: %
right delimiter T: :
BASE C: 4G2G1

Nls: Set Nls (protection for file) Private/Public OK: 4G2H

Private: Set Nls (protection for file) Private OK

The Base command "Set Nls (protection for file) Private" allows you to limit access to the file in which you are currently located. Only those people with their IDENTS listed in the origin statement of the file in exactly the following syntax AccessList: IDENTLIST; will be allowed to see the file. The word AccessList must be followed by a colon (:) and a space before you list the IDENTS. The list of IDENTS must be all capitals, separated by commas, and with a semi-colon (;) at the end. After inserting the AccessList, and setting the protection to private, you must Update the file before the protection will work. If you do not put the AccessList outside the delimiters of the origin statement, it will disappear when you update. NOTE: If you make a mistake, you can easily deny access to yourself. If this occurs, one of your computer's operators should be able to bail you out. See also: origin statement, Set Nls Public, Show File Status. 4G2H1

effects

A NULL access list (i.e., one which is specified but has no IDENTS in it) is very different from a nonexistent AccessList. The former grants access to no one; the latter effectively grants access to everyone. An unauthorized user's attempt to load a private file is

declined by NLS with the message: "Private file: access denied to you." Once a file has been Set Private, it remains private until it is specifically set Public. 4G2H1A

% backlinks: <sendmail, header>, <sendmail, private>

Public: Set Nls (protection for file) Public OK

The Base command "Set Nls (protection for file) Public" makes the file in which you are presently located open to anyone. Set Nls Public undoes what the Set Nls Private command does but is independent of the Set Executive (protection) command. 4G2H2

Temporary: Set Temporary (modifications for file) OK: (really?) OK

The Base command "Set Temporary" allows you to edit a file for your own purposes without having write access (e.g., journal files). The command creates a Partial Copy containing your modifications but does not lock the file you are working in. 4G2I

TNLS example:

BASE C: Set C: Temporary (modifications for file) OK:

BASE C: 4G2I1

write access

##<write>##

modifications

##<nls, modification>##

Reset Temporary (modifications for file) command

##<reset !temporary>##

Viewspecs: Set Viewspecs VIEWSPECS OK

The Base command "Set Viewspecs" allows you to change the viewspecs at any time for the current work session. 4G2J

TNLS example:

BASE C: Set C: Viewspecs V: dmIGy

BASE C: 4G2J1

effects and related commands

To list the viewspecs currently in force for this work session, use the Show Viewspecs command. After this session your viewspecs will revert to a default initial set, which you can select for yourself in the Useroptions subsystem with the Viewspecs and Reset Viewspecs commands. To check what initial set is established, use Useroptions Show Viewpsecs command. The Base Reset Viewspecs command will return current Viewspecs to that same initial set immediately. See also: Base Reset Viewspecs, Base Show Viewspecs, Useroptions Viewspecs, Useroptions Reset Viewspecs. 4G2J2

Window ... 4G2K

Display: Set Window (at) BUG (to) Display (non-printing characters) OK 4G2K1

Reverse: Set Window (at) BUG (to) Reverse (video) OK 4G2K2

Status: Set Window (at) BUG (to) Status (window) OK

The Base command "Set Window Status" allows you to BUG the DNLS window you wish to have simulate a teletype. You will want to use the Break Window command first to create the window. See also: tty-simulation. 4G2K3

set 4G3

insert 4G4

tenex 4G5

Reset

The Base command "Reset" allows you to change back to the original default setting any one of the following things that you specify. See also: Set, Useroptions Reset, Programs Reset, resetting. 4G6

Archive: Reset Archive (request for file) CONTENT OK

The Base command "Reset Archive" allows you to Reset the archival status of all files to the original mode where they will be archived and deleted if they have not been read in 28 days. CONTENT wants you to give the name of an existing file. 4G6A

Case: Reset Case (mode) OK

The Base command "Reset Case" allows you to Reset the Case mode setting for subsequent Force (Case) STRING and Force (Case) STRUCTURE commands to "upper" (all alphabetic characters will be capitalized). This is the default setting. This case mode setting will remain in effect until the Force (Case) Mode command is used again. 4G6B

TNLS example:
BASE C: Reset C: Case (mode) OK:
BASE C: 4G6B1

Content: Reset Content (pattern) OK

The Base command "Reset Content (pattern)" sets aside the content-analyzer you have been using. It is still available to you, but viewspecs i and k will no longer use it to filter statements. The Show Status command in the Programs subsystem will show you a list of content-analyzers that are available to you but not in force at the moment. To enforce one, use the Programs Institute command. 4G6C

TNLS example:
BASE C: Reset C: Content (Pattern) OK:
BASE C: 4G6C1
content-analyzers
##<programs, content-analyzer>##

Link: Reset Link (default for file) OK

The Base command "Reset Link" resets the directory for all links in a file that fail to specify a directory to the default. The default is the directory the file itself resides in. This command is used after the Set Link Default command. If at any time you are not sure which is the link default for your file, use the Show File Default (directory for links) command. 4G6D

TNLS example:
BASE C: Reset C: Link (default for file) OK:
BASE C: 4G6D1

Name: Reset Name (delimiters in) STRUCTURE (at) DESTINATION OK

The Base command "Reset Name" sets the characters that define and mark the boundaries of statement names in a STRUCTURE to your default. See name-delimiters, Set Name, Show Name. 4G6E

TNLS example:
BASE C: Reset C: Name (delimiters in) C: Branch (at) A: 2a
BASE C: 4G6E1

Status: Reset Status (window) OK

The Base command "Reset Status" clears your status simulation window and replaces the file contents that were in the window before you used the Set Status command. See also: Clear. 4G6F

Temporary: Reset Temporary (modifications for file) OK

The Base command "Reset Temporary" erases all the temporary modifications and sets the modification mode back to normal for a given file. You may set the modifications to a file to temporary (with the command Set Temporary modifications) so that you may edit the file for your own purposes without having write access. See also: write. 4G6G

Viewspecs: Reset Viewspecs OK

The Base command "Reset Viewspecs" sets the viewspecs back to your initial set (the default) for work sessions. 4G6H

TNLS example:

BASE C: Reset C: Viewspecs OK:

BASE C: 4G6H1

other commands to manipulate viewspecs

To list the viewspecs currently in force, use the Show Viewspecs status command. One method of changing viewspecs is the Set Viewspecs command. Users can specify their initial set of viewspecs for themselves (the ones you'll go back to when you reset) in the Useroptions subsytem, with that subsytem's Viewspecs command. See also: Base Show Viewspecs, Base Set Viewspecs, Useroptions Viewspecs.

ways of changing viewspecs

##<nls, viewspecs !changing>##

directory-commands: Connect, Expunge, Trim, etc.

The Base commands in this group allow you to connect to other directories in the system and perform certain operations on directories. See also: file-handling commands. 4H

Connect (to): 4H1

Directory: Connect (to) Directory CONTENT (password) PASSWORD OK

The Base command "Connect" allows you to use most NLS and Executive System files as if you had logged in under the directory you specify. If you already have access to the directory you specify for CONTENT, you can just hit <OK> instead of typing the PASSWORD. Ask your computer's operator about directory groups. 4H1A

TNLS example:

BASE C: Connect (to) C: Directory T: guest

(Password) OK/T: T:

BASE C: 4H1A1

effects

Modification files (identified by the extension .PC) containing the changes you have made will appear in the directory where you logged in until you Update the modified file. Also use Connect Directory to return to your own directory after you have connected to another. 4H1A2

modification files

##<nls, modification>##

Display [DISPLAY only]: Connect (to) Display (number) CONTENT (for) Input (and Output).../Output (only) OK

The Base command "Connect (to) Display" allows you to share a display screen with another user's job. Input (and output)

mode is for cooperative work. Output (only) mode does not allow you to do anything except watch. This command only works at Office-1. For detailed information on how to use the command, see the menu below. For Executive System-type linking, use the Connect (to) Tty (for) Output (only). See also: Accept Connect, Disconnect Terminal. 4H1B

To set up the connection

The users who are connecting should be in audio communication (telephone) and must be in agreement about who is giving commands at any particular time. Each needs to know the other's TTY number, which can be found using the TENEX commands "Where", "Systat", or "Jobstat". The user who is going to reach into the other's job gives the "Connect (to) Display " command. Within 15 seconds of his final <CA>, the other user must give the final <CA> to his "Accept Connect " command.

While the connection is in effect

Both users will see the same display. Neither user can see the other's mouse tracking, but he can see the other's bug marks as well as his own. Since anything typed on either terminal will control the job of the user who gave the "Accept" command, both users must be careful not to type at the same time.

To terminate the connection

Give the Base command "Disconnect Terminal <CA>". Either user can do this since both user's input goes to the same job. The user who gave the "Connect" command types <CTRL-P> which restores his input to his own copy of NLS and recreates his screen.

Restrictions

The command only works at OFFICE-1, not at BBNB or ISIC (unless you happen to be a wheel there). For best results, both parties should be at the same kind of terminal. It might work all right if the "Connecting" terminal has a larger screen area than the "Accepting" terminal, but a smaller screen would probably give poor results.

Tty: Connect (to) Tty (number) CONTENT (for) Input.../Output...

Output (only) mode is like the TENEX Link or the TOPS-20 Talk command. Input (and output) mode works like the Executive System ADVISE command. 4H1C

Output: Connect (to) Tty (number) CONTENT (for) Output (only) OK

The Base command "Connect (to) Tty (number)" makes what you type appear at the other terminal and vise versa. To find someone's terminal number, use the WHERE or SYSTAT commands in TENEX or the SYSTAT command in TOPS-20. Follow the "Connect" command with the semicolon command. <CTRL-P>, <CTRL-O> or the Disconnect command will break the connection. See also: semicolon, Disconnect, work stations, connecting. 4H1C1

DNLS useage

The DNLS user may communicate with TNLS users and other DNLS users with Base's Connect to Tty command. Once connected, use the tty window to see what the other user is doing, or go to the Executive System level. You

will be able to see anything a TNLS user is doing while you are linked to him. The only actions of another DNLS user you will see is whatever would appear in his tty window or at his Executive System level. This is enough if you intend just to comment (Semicolon command). To see everything a DNLS user does, i.e., share viewing a file (but not cursors), use Base's Connect to Display command. See also: connecting, Connect Tty, Connect Display, tty-simulation window, cursor, semicolon.
TNLS example:
BASE C: Connect (to) C: Tty (Number) T/[A]: 5
(for) C: Output (Only) OK:
BASE C: 4H1C1B

CONTENT wants the user's tty number.
(This is the number the system assigns to each terminal). Warning: Connecting Tty's might affect the other user's job if she's in NDNLS, OUTPRC, L10, or other special subsystems; EXEC (TENEX or TOPS-20), TNLS, NTNLS, or DNLS are usually safe. Use the Where command in TENEX or the Systat command in TOPS-20 for all this information about another user.

Input: [TYPEWRITER only] Connect (to) Tty (number) CONTENT (for) Input (and output) OK

If you specify the Input (and output) mode, you will be advising the job at the terminal number you specify for CONTENT. This means the person to whom you are connecting must issue the "Accept Connection" command and any commands you give will affect the job of the person you are linked to, not your own job. For conversation, use the Output (only) mode. Do not use this command in DNLS, use the Connect (to) Display command instead. To find someone's terminal number, use the WHERE or SYSTAT commands in TENEX or the SYSTAT command in TOPS-20. <CTRL-P> or the Disconnect command will break the connection. See also: semicolon, Disconnect, terminals, connecting. 4H1C2

TNLS example:
BASE C: Connect (to) C: Tty (Number) T: 27
(for) C: Input (and Output) OK:
BASE C: 4H1C2A

Expunge

The Base command "Expunge" completely erases deleted files from the running system. Deleted files must be expunged before their disk pages are free. After files have been expunged they cannot be undeleted or retrieved, unless you try to expunge the file in which your current marker resides. Occasionally if the entire system is low on disk pages, it will automatically expunge your deleted files. See also: Undelete. 4H2

Directory: Expunge (deleted files from) Directory OK

The Base command "Expunge Directory" permanently erases from the system all deleted files in the directory you logged in under or subsequently connected to (unless the files are being held open by someone using them). See Expunge. See also: Connect Directory, Delete File, Undelete File. 4H2A

TNLS example:
BASE C: Expunge (deleted files from) C: Directory OK:
BASE C: 4H2A1

Reset Link Default
##<reset !link>##
Set Link Default
##<set !link>##
Show File Default (directory for links):
##<show !file !default>##
Show Disk (space status)
##<show !disk>##
Trim Directory (no. versions to keep) CONTENT OK (really?) OK
The Base command "Trim Directory" will delete any extra versions of each file so that there are no more than the given number of versions. If you have write access to the directory (either by definition or having connected to it), you don't need to type the password. If you don't specify a directory, it will trim the directory to which you are connected; the password won't be necessary. 4H7
TNLS example:
BASE C: Trim C: Directory (no. versions to keep) T: 1
(really?) OK:
Trimmed Files are
(DOE, WONDER.NLS,) 2 files deleted
BASE C: 4H7A
versions
##<nls, version>##
write access
##<write>##
terminal-commands: Simulate, Disconnect, Accept Connect, etc.
The Base commands in this group allow you to control certain terminal functions. See also: directory-commands. 4I
Accept Connect (from terminal number) (for) Input (and Output).../Output (Only) CONTENT OK
The Base command "Accept Connect (from terminal number)" allows someone at the terminal number you specify for CONTENT to share your job with you. After you have been connected to, you have 15 seconds to accept the connection. "Accept Connect (for) Input" allows the person connecting to you to execute commands on your job. "Accept Connect (for) Output" only allows viewing. You must know ahead of time which Connect command the other person is using. You should be in audio communication with the person connecting to you to make this command work effectively. See also: Disconnect Terminal, Connect Tty. 4I1
TNLS example:
BASE C: Accept C: Connect (from terminal number) T/[A]: 5
(for) C: Output (Only) OK:
BASE C: 4I1A
instructions for sharing screen via Connect Display/Accept Connect
##<connect !display>##
Disconnect Terminal OK
The Base command "Disconnect Terminal" breaks a connection with another terminal that either user has established with the "Connect Display" command or "Connect Tty" command. The "Disconnect Terminal" command must be followed by <CTRL-P> to restore the screens of both terminals. See also: Connect Display, Connect Tty, linking. 4I2
TNLS example:

BASE C: Disconnect C: Terminal OK:
BASE C: 4I2A

Simulate...
The Simulate commands enable you to go from DNLS to TNLS and back again when you are working at a display terminal. These commands only work from a display terminal. 4I3

Display: Simulate Display OK
The TNLS command "Simulate Display" returns you to DNLS after you have used the Simulate Typewriter command at a display terminal. 4I3A

Typewriter: Simulate Typewriter OK
The Base command "Simulate Typewriter" makes your display terminal mimic a typewriter terminal, preserving the state of the current work session as much as possible. Its common use is to move a display user DNLS to TNLS. Use the command Simulate Display to return to DNLS from TNLS. 4I3B

Connect to Tty
##<connect !tty>##
Connect to Display
##<connect !display>##
Clear (status window)
##<clear>##
semicolon typed at the herald
##<nls, universal !semicolon>##

display-commands only: Clear, Freeze Statement, Release, etc.
The Base group of display-only commands allows you to control certain functions of your display screen and alter your view of information. See also: terminal-commands, viewspecs. 4J

Accept Connect
##<accept>##
Connect to Display
##<connect !display>##
Delete Edge
an NLS 8.5 command that is now replaced by
##<delete !window>##
Delete Window
##<delete !window>##
Insert Edge
an NLS 8.5 command that is now replaced by
##<break !window>##
Jump commands in all subsystems
##<nls, jump>##
Move Edge
an NLS 8.5 command that is now replaced by
##<enlarge>##
Reset Status (window)
##<reset !status>##
Set Status (window)
##<set !status>##
%uparrow for DNLS
##<nls, uparrow>##
Break Window
##<break !window>## 4J11

Clear (status window) OK:
The Base (display only) command "Clear (status window)" clears your status window. See also: status. 4J12

Enlarge Window
##<enlarge>## 4J13
Freeze Statement (at) DESTINATION VIEWSPECS OK
The DNLS command "Freeze" keeps the statement at the DESTINATION you specify on the upper part of the screen (when viewspec o is on). You may continue to use other NLS commands in the lower part of your screen. For the VIEWSPECS field specify viewspecs to define the format of the frozen statement. You cannot use the field to turn on viewspec o; you must do that separately. If you freeze more than one statement, they all will show when you turn viewspec o on. The Release command "thaws" frozen statements. See also: viewspec o, display-control viewspecs. 4J14
Release
The Base (DNLS only) command "Release" thaws statements that were frozen at the top of your screen with the "Freeze" command. 4J15
All: Release All (frozen statements) OK
The Base (DNLS only) command "Release All" thaws all statements that were frozen with the "Freeze" command. 4J15A
Frozen: Release Frozen (statement at) DESTINATION OK
The Base (DNLS only) command "Release Frozen" thaws the statement you specify for DESTINATION that was frozen with the "Freeze" command. 4J15B

special-purpose commands: Process, Mark Character, record commands
The Base commands in this group allow you to do a variety of special purpose tasks. 4K
Process (commands from) STRUCTURE OK
The Base command "Process" carries out a procedure that involves a series of commands without you having to specify each command separately. The commands you want executed must be written for DEMAND recognition mode. Insert CTRL-characters such as <CTRL-D> <CA> or <CTRL-E> by using the literal escape key <CTRL-V> before typing the CTRL-character. Because the process that carries out commands skips over the name of a statement, many users begin this branch with a statement consisting only of a name and place the actual commands in the branch under the name. If your name delimiters delimiters are NULL NULL place a space at the beginning of each command to avoid having the process skip over the first word. Note that DEMAND recognition mode requires a space after each commandword even if it is followed by <CA>. You can have the system automatically process commands every time you login by using the Useroptions Startup command. 4K1
% backlinks: <sendmail, process>
Playback Record (of session from file) CONTENT (simulate recorded timing?) ANSWER OK
The Base command "Playback Record" will pass control to the file you recorded on and precisely the same commands will be seen and executed on the file you were working on as during the control session. This will happen in approximately the same timing if you ANSWER with a y (for yes). If you ANSWER with an n (for no), the session will playback as fast as it can. CONTENT wants you to TYPEIN or point to the FILEADDRESS of the file you recorded on. You should be careful that everything is set up properly when you playback the session. If you get into trouble,

<CTRL-O> stops the playback. 4K2

TNLS example:
BASE C: Playback C: Record (of session from file) T: rf
(simulate recorded timing?) Y/N: OK:
[commands recorded on file will be executed here]
BASE C: 4K2A

Start Record command
##<start>##
Start Record (of session to file) CONTENT OK
The Base command "Start Record" allows you to begin to record on a file all the interactions with NLS on another file. CONTENT wants you to TYPEIN or point to a FILEADDRESS of the file on which you want to record all the things you do. To terminate the recording, use the command Stop Record of session. Then you can see a simulation of exactly what you did by using the command Playback. 4K3

Stop Record (of session) OK
The Base command "Stop Record" stops recording your NLS session on the file you had designated to record your interactions. At that point, the file will be closed and no more commands will be recorded. See Start Record, Playback Record. 4K4

TNLS example:
BASE C: Stop C: Record (of session) OK:
BASE C: 4K4A

Mark Character (at) DESTINATION (with marker named) CONTENT OK
The Base command "Mark Character" assigns a marker at the DESTINATION you specify in the file, with the name you specify as CONTENT. See also: Show Marker, Delete Marker, Delete All. 4K5

TNLS example:
BASE C: Mark C: Character (at) A: 1a
(with marker named) T/[A]: tiger
BASE C: 4K5A
markers
##<nls, marker>## 4K5B

%

%Reading Grade Levelomitting statements less than 70 characters=7.66
% syntax of base commands % 6

Load Program CONTENT CONFIRM
Compile File (at) DESTINATION (using) CONTENT (to file) CONTENT CONFIRM
Compile Procedure (at) DESTINATION CONFIRM
Compile L10 (user program at) DESTINATION CONFIRM
Compile Content (pattern) CONTENT CONFIRM
Verify File CONFIRM
Update (file) New CONFIRM
Update (file) Old (version) CONFIRM
Update (file) Compact CONFIRM
Update (file) Rename (to filename) CONTENT CONFIRM
^

Transpose String (at) DESTINATION (and) DESTINATION OPTION (filtered:) CONFIRM
Transpose String (at) DESTINATION (and) DESTINATION CONFIRM
Transpose Structure (at) DESTINATION (and) DESTINATION OPTION (filtered:) CONFIRM
Transpose Structure (at) DESTINATION (and) DESTINATION CONFIRM

Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) OPTION (filtered:) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT (for all occurrences of the old) CONTENT (finished?) Status CONFIRM (finished?) ANSWER (the new) CONTENT (for all occurrences of the

```
old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT
(for all occurrences of the old) CONTENT (finished?) Status CONFIRM
(finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT
(for all occurrences of the old) CONTENT (finished?) Status CONFIRM
(finished?) ANSWER (the new) CONTENT (for all occurrences of the
old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT
(for all occurrences of the old) CONTENT (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT
(for all occurrences of the old) CONTENT (finished?) ANSWER (the
new) CONTENT (for all occurrences of the old) CONTENT (finished?)
Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT
(for all occurrences of the old) CONTENT (finished?) ANSWER (the
new) CONTENT (for all occurrences of the old) CONTENT (finished?)
Status CONFIRM (finished?) ANSWER CONFIRM
Substitute String (in) Structure (at) DESTINATION (the new) CONTENT
(for all occurrences of the old) CONTENT (finished?) ANSWER (the
new) CONTENT (for all occurrences of the old) CONTENT (finished?)
ANSWER CONFIRM
Sort Group DESTINATION CONFIRM
Sort Branch DESTINATION CONFIRM
Sort Plex DESTINATION CONFIRM
/
Show File Status CONFIRM
Show File Default (directory for links) CONFIRM
Show File Modifications (status) CONFIRM
Show File Return (ring) CONFIRM
Show File Size CONFIRM
Show Return (ring) CONFIRM
Show Marker (list) CONFIRM
Show Name (delimiters for statement at) DESTINATION CONFIRM
Show Viewspecs (status) Verbose CONFIRM
Show Viewspecs (status) CONFIRM
Set External (names link file to:) CONTENT CONFIRM
Set Content (pattern) To CONTENT CONFIRM
Set Content (pattern) On CONFIRM
Set Content (pattern) Off CONFIRM
Set Link (default for file to directory) CONTENT CONFIRM
Set Name (delimiters in) Structure (at) DESTINATION (left delimiter)
CONTENT (right delimiter) CONTENT CONFIRM
Set Nls (protection for file) Private CONFIRM
Set Nls (protection for file) Public CONFIRM
Set Temporary (modifications for file) CONFIRM (really?) CONFIRM
Set Tty (simulation for window) DESTINATION CONFIRM
Set Viewspecs CONFIRM
Reset Case (mode) CONFIRM
Reset Content (Pattern) CONFIRM
Reset Link (default for file) CONFIRM
Reset Name (delimiters in) Structure (at) DESTINATION CONFIRM
Reset Temporary (modifications for file) CONFIRM
Reset Tty (window) CONFIRM
Reset Viewspecs CONFIRM
Replace String (at) DESTINATION (by) CONTENT CONFIRM
```

```
Replace Structure (at) DESTINATION (by) CONTENT CONFIRM
Renumber Sids (in file) CONFIRM
Release Frozen (statement at) DESTINATION CONFIRM
Release All (frozen statements) CONFIRM
Print Rest CONFIRM
Print File CONFIRM
Print Structure (at) DESTINATION
Print Journal (mail) CONFIRM
.
Output (to) Printer Append (to file) CONTENT CONFIRM
Output (to) Printer File CONTENT CONFIRM
Output (to) Printer Copies CONTENT CONFIRM
Output (to) Printer CONFIRM
Output (to) Com (for device) Singer CONFIRM Append (to file) CONTENT
CONFIRM
Output (to) Com (for device) Singer CONFIRM File CONTENT CONFIRM
Output (to) Com (for device) Singer CONFIRM Copies CONTENT CONFIRM
Output (to) Com (for device) Singer CONFIRM Test File CONTENT
CONFIRM
Output (to) Com (for device) Singer CONFIRM CONFIRM
Output (to) Com (for device) Comp80 CONFIRM Append (to file) CONTENT
CONFIRM
Output (to) Com (for device) Comp80 CONFIRM File CONTENT CONFIRM
Output (to) Com (for device) Comp80 CONFIRM Copies CONTENT CONFIRM
Output (to) Com (for device) Comp80 CONFIRM Test File CONTENT
CONFIRM
Output (to) Com (for device) Comp80 CONFIRM CONFIRM
Output (to) Com (for device) CONFIRM Append (to file) CONTENT
CONFIRM
Output (to) Com (for device) CONFIRM File CONTENT CONFIRM
Output (to) Com (for device) CONFIRM Copies CONTENT CONFIRM
Output (to) Com (for device) CONFIRM Test File CONTENT CONFIRM
Output (to) Com (for device) CONFIRM CONFIRM
Output (to) Quickprint No (headers) Append (to file) CONTENT CONFIRM

Output (to) Quickprint No (headers) File CONTENT CONFIRM
Output (to) Quickprint No (headers) Copies CONTENT CONFIRM
Output (to) Quickprint No (headers) CONFIRM
Output (to) Quickprint Append (to file) CONTENT CONFIRM
Output (to) Quickprint File CONTENT CONFIRM
Output (to) Quickprint Copies CONTENT CONFIRM
Output (to) Quickprint CONFIRM
Output (to) Journal (quickprint) No (headers) Append (to file)
CONTENT CONFIRM
Output (to) Journal (quickprint) No (headers) File CONTENT CONFIRM
Output (to) Journal (quickprint) No (headers) Copies CONTENT CONFIRM

Output (to) Journal (quickprint) No (headers) CONFIRM
Output (to) Journal (quickprint) Append (to File) CONTENT CONFIRM
Output (to) Journal (quickprint) File CONTENT CONFIRM
Output (to) Journal (quickprint) Copies CONTENT CONFIRM
Output (to) Journal (quickprint) CONFIRM
Output (to) Sequential Append (to file) CONTENT Force (upper case)
CONFIRM
Output (to) Sequential Append (to file) CONTENT CONFIRM
Output (to) Sequential File CONTENT Force (upper case) CONFIRM
```

Output (to) Sequential File CONTENT CONFIRM
Output (to) Assembler Append (to file) CONTENT Force (upper case) CONFIRM
Output (to) Assembler Append (to file) CONTENT CONFIRM
Output (to) Assembler File CONTENT Force (upper case) CONFIRM
Output (to) Assembler File CONTENT CONFIRM
Output (to) Terminal File CONTENT CONFIRM (send form feeds?) ANSWER (wait at page break?) ANSWER (CONFIRM when ready) CONFIRM
Output (to) Terminal File CONTENT CONFIRM (send form feeds?) ANSWER (simulate?) ANSWER (wait at page break?) ANSWER (CONFIRM when ready) CONFIRM
Output (to) Terminal CONFIRM (send form feeds?) ANSWER (wait at page break?) ANSWER (CONFIRM when ready) CONFIRM
Output (to) Terminal CONFIRM (send form feeds?) ANSWER (simulate?) ANSWER (wait at page break?) ANSWER (CONFIRM when ready) CONFIRM
Output (to) Remote (printer -- TIP) CONTENT (Port #) CONTENT CONFIRM (send form feeds?) ANSWER (wait at page break?) ANSWER (CONFIRM when ready) CONFIRM
Output (to) Remote (printer -- TIP) CONTENT (Port #) CONTENT CONFIRM (send form feeds?) ANSWER (Simulate?) ANSWER (wait at page break?) ANSWER (CONFIRM when ready) CONFIRM
Move Link (from) SOURCE (to follow) DESTINATION CONFIRM

Move Text (from) SOURCE (to follow) DESTINATION CONFIRM
Move Structure (from) SOURCE (to follow) DESTINATION OPTION (filtered:) CONFIRM
Move Structure (from) SOURCE (to follow) DESTINATION CONFIRM
Merge Group (at) SOURCE (into) DESTINATION CONFIRM

Merge Plex (at) SOURCE (into) DESTINATION CONFIRM
Mark Character (at) DESTINATION (with marker named) CONTENT CONFIRM <LF>
Insert Link (to follow) DESTINATION CONTENT CONFIRM
Insert Number (to follow) DESTINATION CONTENT CONFIRM
Insert String (to follow) DESTINATION CONTENT CONFIRM
Insert Text (to follow) DESTINATION CONTENT CONFIRM
Insert Structure (to follow) DESTINATION CONTENT CONFIRM
Insert Structure (to follow) DESTINATION CONTENT CONFIRM
Insert Date (to follow) DESTINATION CONFIRM
Insert Time (and date to follow) DESTINATION CONFIRM
Insert Sendmail (form) (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Case (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Case (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Catchphrase (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Catchphrase (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Coroutine (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Coroutine (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Do-until (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Do-until (to follow) DESTINATION CONFIRM

Insert Programmer's (element) Do-while (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Do-while (to follow) DESTINATION CONFIRM
Insert Programmer's (element) For (to follow) DESTINATION CONFIRM
Insert Programmer's (element) For (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Grammar (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Grammar (to follow) DESTINATION CONFIRM
Insert Programmer's (element) If-then-else (to follow) DESTINATION CONFIRM
Insert Programmer's (element) If-then-else (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Loop (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Loop (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Parsefunction (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Parsefunction (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Procedure (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Procedure (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Until-do (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Until-do (to follow) DESTINATION CONFIRM
Insert Programmer's (element) While-do (to follow) DESTINATION CONFIRM
Insert Programmer's (element) While-do (to follow) DESTINATION CONFIRM
Insert Programmer's (element) Comment (to follow) DESTINATION CONTENT CONFIRM
Freeze Statement (at) DESTINATION CONFIRM
Force (case) String (at) DESTINATION Upper CONFIRM
Force (case) String (at) DESTINATION Lower CONFIRM
Force (case) String (at) DESTINATION First (letter upper) CONFIRM
Force (case) String (at) DESTINATION CONFIRM
Force (case) Structure (at) DESTINATION Upper CONFIRM
Force (case) Structure (at) DESTINATION Lower CONFIRM
Force (case) Structure (at) DESTINATION First (letter upper) CONFIRM

Force (case) Structure (at) DESTINATION CONFIRM
Force (case) Mode Upper CONFIRM
Force (case) Mode Lower CONFIRM
Force (case) Mode First (letter upper) CONFIRM
Expand (window) DESTINATION (to) DESTINATION CONFIRM
Edit Statement (at) DESTINATION
Delete String (at) DESTINATION CONFIRM
Delete Structure (at) DESTINATION OPTION (filtered:) CONFIRM
Delete Structure (at) DESTINATION CONFIRM
Delete Marker (named) (not implemented) CONTENT CONFIRM
Delete All (markers) (not implemented) CONFIRM
Delete Modifications (to file) CONFIRM (really?) CONFIRM
Delete File CONTENT CONFIRM

Delete Window DESTINATION (expanding window) DESTINATION CONFIRM
Create File CONTENT CONFIRM
Copy Link (from) SOURCE (to follow) DESTINATION CONFIRM
Copy Number (from) SOURCE (to follow) DESTINATION CONFIRM
Copy String (from) SOURCE (to follow) DESTINATION CONFIRM
Copy Text (from) SOURCE (to follow) DESTINATION CONFIRM
Copy Structure (from) SOURCE (to follow) DESTINATION OPTION
(filtered:) CONFIRM
Copy Structure (from) SOURCE (to follow) DESTINATION CONFIRM
Copy Sequential (file from) CONTENT (to follow) DESTINATION (using)
One (<CR> to end statement) CONFIRM
Copy Sequential (file from) CONTENT (to follow) DESTINATION (using)
Two (<CR>s end statement) Justified (delete extra <SP>) CONFIRM
Copy Sequential (file from) CONTENT (to follow) DESTINATION (using)
Two (<CR>s end statement) CONFIRM
Copy Sequential (file from) CONTENT (to follow) DESTINATION (using)
Assembler CONFIRM
Connect (to) Directory CONTENT (password) CONTENT CONFIRM
Connect (to) Directory CONTENT (password) CONFIRM
Clear (status window) CONFIRM
\
Break Statement (at) DESTINATION CONFIRM OPTION (with inserted text)
CONTENT CONFIRM
Break Statement (at) DESTINATION CONFIRM
Break Window Vertically (at) Center (of window) DESTINATION
(displaying in) DESTINATION CONFIRM
Break Window Vertically (at) DESTINATION (displaying in) DESTINATION
CONFIRM
Break Window Horizontally (at) Center (of window) DESTINATION
(displaying in) DESTINATION CONFIRM
Break Window Horizontally (at) DESTINATION (displaying in)
DESTINATION CONFIRM
Append Statement (at) SOURCE (to) DESTINATION (join with) CONTENT
CONFIRM
Quit CONFIRM
Jump (to) DESTINATION CONFIRM
Jump (to) Address (relative to) DESTINATION (modified by address)
CONTENT CONFIRM
Jump (to) Jump (to) Address DESTINATION CONFIRM
Jump (to) Item DESTINATION CONFIRM
Jump (to) Successor DESTINATION CONFIRM
Jump (to) Predecessor DESTINATION CONFIRM
Jump (to) Up DESTINATION CONFIRM
Jump (to) Down DESTINATION CONFIRM
Jump (to) Head DESTINATION CONFIRM
Jump (to) Tail DESTINATION CONFIRM
Jump (to) End (of Branch) DESTINATION CONFIRM
Jump (to) Back DESTINATION CONFIRM
Jump (to) Origin DESTINATION CONFIRM
Jump (to) Next DESTINATION CONFIRM
Jump (to) Link CONTENT CONFIRM
Jump (to) Name Any CONTENT CONFIRM
Jump (to) Name First CONTENT CONFIRM
Jump (to) Name Next CONTENT CONFIRM
Jump (to) Name External CONTENT CONFIRM
Jump (to) Return CONFIRM ANSWER

```
Jump (to) Return CONFIRM ANSWER ANSWER
Jump (to) File DESTINATION CONFIRM
Jump (to) File Named CONTENT CONFIRM
Jump (to) File Return CONFIRM ANSWER
Jump (to) File Return CONFIRM ANSWER ANSWER
Jump (to) Content First       CONFIRM
Jump (to) Content First CONTENT CONFIRM
Jump (to) Content First       CONFIRM
Jump (to) Content First CONTENT CONFIRM
Jump (to) Content Next        CONFIRM
Jump (to) Content Next CONTENT CONFIRM
Jump (to) Content Next        CONFIRM
Jump (to) Content Next CONTENT CONFIRM
Jump (to) Word First          CONFIRM
Jump (to) Word First CONTENT CONFIRM
Jump (to) Word First          CONFIRM
Jump (to) Word First CONTENT CONFIRM
Jump (to) Word Next   CONFIRM
Jump (to) Word Next CONTENT CONFIRM
Jump (to) Word Next   CONFIRM
Jump (to) Word Next CONTENT CONFIRM
Handle CONFIRM
Goto (subsystem) CONTENT CONFIRM
Goto (subsystem) OPTION (subsystem name) CONTENT CONFIRM

% source-code for base subsystem                                    7
  %This is the BASE grammar for the NLS 9 system, outside the NSW,
  using a shared page protocol between the Frontend and the Backend%
  % grammar code %                                                  7B
    FILE cbase % using  [from NLS10] (arcsubsys,cml10,) to
    (RELNINE,base.cgr,) %                                          7B1
      % note that this base grammar includes the programmer's editor
      commands and rules as well as the programs commands compile
      and load program because the switch PROGRAMMERS=TRUE.  In the
      future for programmer's editor compile to
      (relnine,ptools.cml,) %
    % FLAGS %                                                      7B2
      INCLUDE <nine, nls-grammar, flags !shared>                  7B2A
      INCLUDE <nine, nls-grammar, flags !base>                    7B2B
      SET PROGRAMMERS=FALSE;  %TRUE for programmer's toolkit%
      SET XVER=FALSE;  %TRUE for experimental grammar%
    % DECLARATIONS %                                              7B3
      INCLUDE <nine, nls-grammar, declarations !universal>        7B3A
      INCLUDE <nine, nls-grammar, declarations !base>             7B3B
    % RULES COMMONLY USED %                                       7B4
      INCLUDE <nine, nls-grammar, rules !universal>               7B4A
      %+PROGRAMMERS%                                              7B4B
      INCLUDE <nine, nls-grammar, rules !programmer>              7B4C
      %+PROGRAMMERS%                                              7B4D
    % COMMANDS % SUBSYSTEM base KEYWORD "BASE"                    7B5
      % BASE COMMANDS %
        INCLUDE <nine, nls-grammar, commands !base>              7B5A1
      % PROGRAMMERS COMMANDS %
        %+PROGRAMMERS%                                           7B5B1
        INCLUDE <nine, nls-grammar, commands !programmer>        7B5B2
        %+PROGRAMMERS%                                           7B5B3
```

```
      % UNIVERSAL COMMANDS  %
         INCLUDE <nine, nls-grammar, commands !universal>        7B5C1
   END.
   FINISH OF BASE                                                  7B7
% parse function source code %                                     7C
   FILE basepfc % <arcsubsys, x110,> <arcsubsys, 1109,> to
   <relnine,base.pfc,> <relnine,testbase.pfc,>%
   INCLUDE < nine, nls-parse, declarations >
   INCLUDE < nine, nls-parse, code >
   FINISH
```

% be code: see <nls-library, be: ebt> % 7D

```
12345678901234567890123456789012345678901234567890123456789012345678901234567
 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor SI


  SSSSSSSS     HH       HH     EEEEEEEEEE     RRRRRRRR       WW       WW       00000
  SSSSSSSS     HH       HH     EEEEEEEEEE     RRRRRRRR       WW       WW       00000
SS             HH       HH     EE             RR      RR     WW       WW     00
SS             HH       HH     EE             RR      RR     WW       WW     00
SS             HH       HH     EE             RR      RR     WW       WW     00
SS             HH       HH     EE             RR      RR     WW       WW     00
  SSSSSS       HHHHHHHHHH     EEEEEEEE       RRRRRRRR       WW       WW     00
  SSSSSS       HHHHHHHHHH     EEEEEEEE       RRRRRRRR       WW       WW     00
        SS     HH       HH     EE             RR   RR        WW   WW  WW     00
        SS     HH       HH     EE             RR   RR        WW   WW  WW     00
        SS     HH       HH     EE             RR    RR       WWWW  WWWW     00
        SS     HH       HH     EE             RR    RR       WWWW  WWWW     00
SSSSSSSS       HH       HH     EEEEEEEEEE     RR      RR     WW       WW       0000
SSSSSSSS       HH       HH     EEEEEEEEEE     RR      RR     WW       WW       0000


 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor S

 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor S

 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor S

 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor S

 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor S

 *START* User SHERWOOD  Job  FTP Seq. 4442 Date 26-Jul-78 07:19:27 Monitor S
```

(acsaverr)	<nine, calculator, 01351>	STRING	6C2B6
(badaccno)	<nine, calculator, 01348>	STRING	6C2B3
(badcfile)	<nine, calculator, 01352>	STRING	6C2B7
(blanks)	<nine, calculator, 01356>	STRING	6C2B8
(caclear)	<nine, calculator, 0805>	PROCEDURE	6C5A
(cafilinitialize)	<nine, calculator, 0782>	PROCEDURE	6C4A
(calculator)	<nine, calculator, 01321>	EXT	6C2C
(calsyserr)	<nine, calculator, 01350>	STRING	6C2B5
(caupderr)	<nine, calculator, 01349>	STRING	6C2B4
(ceval)	<nine, calculator, 02178>	PROCEDURE	6C3C2
(cevend)	<nine, calculator, 02591>	PROCEDURE	6C3C3
(cfeedback)	<nine, calculator, 084>	PROCEDURE	6C3B1
(formdigerr)	<nine, calculator, 01347>	STRING	6C2B2
(opconvert)	<nine, calculator, 02587>	PROCEDURE	6C3C4
(spliterr)	<nine, calculator, 01346>	STRING	6C2B1
(xcalcinit)	<nine, calculator, 042>	PROCEDURE	6C3A1
(xcarith)	<nine, calculator, 01077>	PROCEDURE	6C3C1
(xceval)	<nine, calculator, 0206>	PROCEDURE	6C3F1
(xcevend)	<nine, calculator, 0276>	PROCEDURE	6C3F2
(xcfeedb)	<nine, calculator, 0420>	PROCEDURE	6C3G5
(xcinsert)	<nine, calculator, 02202>	PROCEDURE	6C3N1
(xcjust)	<nine, calculator, 0363>	PROCEDURE	6C3G2
(xciraccum)	<nine, calculator, 0171>	PROCEDURE	6C3D1
(xclrfil)	<nine, calculator, 0189>	PROCEDURE	6C3E1
(xcomma)	<nine, calculator, 0394>	PROCEDURE	6C3G3
(xcquit)	<nine, calculator, 01796>	PROCEDURE	6C3Q1
(xcreenter)	<nine, calculator, 0676>	PROCEDURE	6C3P1
(xcreplace)	<nine, calculator, 02243>	PROCEDURE	6C3O1
(xcshoaccums)	<nine, calculator, 0434>	PROCEDURE	6C3H1
(xcshofil)	<nine, calculator, 0459>	PROCEDURE	6C3I1
(xctotl)	<nine, calculator, 0489>	PROCEDURE	6C3J1
(xcuseaccum)	<nine, calculator, 0506>	PROCEDURE	6C3K1
(xcusesaved)	<nine, calculator, 0979>	PROCEDURE	6C3L1
(xcwritef)	<nine, calculator, 0629>	PROCEDURE	6C3M1
(xdollar)	<nine, calculator, 0407>	PROCEDURE	6C3G4
(xfdigits)	<nine, calculator, 0313>	PROCEDURE	6C3G1

Calculator subsystem

The Calculator subsystem provides a variety of commands that allow you to do simple arithmetic--add, subtract, multiply, and divide--and integrate your totals into an NLS file. To load Calculator (unless Calculator has been made automatically available to you with the Useroptions Include command), use the Goto command, type the option character <CTRL-U>, and type "calculator". Once Calculator has been loaded during a work session, simply give the command Goto Calculator.

% Index in <nls,>
% FIRST SEARCHES

accumulators
##<accumulator>## 2A

calcfile
##<calculator-file>## 2B

calc-file
##<calculator-file>## 2C

calc-ident
##<calculator-file>## 2D

file
##<calculator-file>## 2E

operator
##<evaluate>## 2F

How to use the Calculator subsystem

When you add, subtract, multiply, and divide with the Calculator subsystem, a special "Calculator-File" automatically records your running total, the numbers you use, and the operations you perform. You can have up to ten running totals at one time; each total is stored in a numbered "accumulator." You can insert any accumulator total to follow any string or structure you designate in an NLS file, or replace any string or structure you designate in an NLS file with your total. Once you go to the Calculator subsystem, the prompt NUM/C: lets you know that the calculator is waiting for a command. NUM: prompts for a number, C: for a command word. 3

entering numbers

You can enter numbers in the Calculator subsystem in the following ways:

- typing them in;
- giving an address of a number;
- bugging in DNLS;
- specifying an accumulator where a number is stored.

You can begin performing an arithmetic operation when you are prompted by Num/C: 3A

numbers recognizable to the Calculator

123456	-123456	123456-
123.12	$123.00	0.12345-
123,456	($1,123,123)	12,123.123+
0.1	.12-	0.11
+1		

3A1

formatting numbers with dollar signs, commas, and decimal points

You can specify a format for your numbers. You can indicate whether or not you want commas and/or dollar signs and where you want your decimal point to be. The Calculator subsystem has internal checks that prevent you from inputting numbers that do not fit your format or formats that cannot be used for numbers that already exist. Format numbers with dollar signs, commas,

and change the location of the decimal point with the following commands: Format Commas, Format Dollar, Format Left, Format Right, Format Places. See format. 3A2

Calculator-File

When you first use the Calculator subsystem, a file named "CALC-IDENT.NLS" (your ident) is created in your directory. Subsequently, this file is automatically loaded each time you use the Calculator tool. "CALC-IDENT" records the history of your work like the tape on an adding machine. It keeps records of each number you enter, the operations you perform, subtotals, and totals. All items are first level statements. A line of asterisks marks the beginning of each session. Your "CALC-IDENT" file is a standard NLS file and may be printed. It should not be edited, however. The Calculator command "Write" copies this file to another NLS file which may be edited. (The Base Copy command may also be used to copy all or portions of the file.) 3B

accumulator

When using the Calculator, your running totals are stored in accumulators. The number of the accumulator you are using appears in your Tty window. When you begin a calculator session, accumulator 1 is automatically loaded (although the number is not shown at this point). Each time you begin a calculator session, each of the ten accumulators is set to 0.

You may specify a particular accumulator with the command "Use Accumulator" and a number from one to ten. Your Tty window will display the number and total of the specified accumulator. If you have already used this accumulator in this session, its last total will be shown; if you have not used it, the total will be 0.

You may reset to 0 the value in the accumulator you are using with the command "Clear Accumulator". The accumulators you were using when you ended your last calculator session are also available. 3C

using saved accumulators

You may use accumulators from your previous session, but once you request a saved accumulator, you no longer have the accumulators from your present session available to you. The following example illustrates how this works:

Row A represents five accumulator totals you are using in your present session. Row B represents those from your previous session.

Row A Acc.1 11, Acc.2 22, Acc.3 33, Acc.4 44, Acc.5 55.

Row B Acc.1 66, Acc.2 77, Acc.3 88, Acc.4 99, Acc.5 95.

You are using accumulator 3 (Row A, value 33) and give the "Use Saved" command, which furnishes you with accumulator 3 from Row B (value 88). If you then give the command "Use Accumulator" and specify number 2, you will be furnished with accumulator 2 from Row B (value 77). You no longer have access to any of the values in the accumulators from Row A. 3C1

arithmetic: doing simple problems

Simple arithmetic operations may be performed with the Add, Divide, Evaluate, Multiply, Subtract, and Total commands. Here is one way to do a problem:

1. At the herald, give a command: either one of the command words Add, Subtract, Multiply, or Divide OR one of the signs +, -, x, *, or /. (If you omit a command, addition is assumed.)

2. Follow the command word (or sign) by a number and an <OK>. (See numbers.)
3. The number you gave and its command enter a special file.
4. The designated operation is performed on your number and the total in the accumulator you are using. The answer replaces the old value in the accumulator you are using and you are ready to begin your next command. (See totals.) 3D

totals: using them in two ways

Make use of other totals from your present Calculator session or the most recent Calculator session, and erase totals you have stored with the following commands: Use Accumulator, Use Saved, Clear Accumulator, Clear File.

The total you have in an accumulator can be inserted in an NLS file, or can replace any string or structure you designate in an NLS file. Write a total in another NLS file or edit the file that contains the record of your calculations with the follwing commands: Insert Accumulator, Write File. 3E

The Calculator subsystem can be useful in a variety of ways.

It can help in planning and preparing budget reports, filling out forms that involve numbers such as an income tax form, and doing simple arithmetic procedures where the numbers keep changing.

Commands in the Calculator subsystem 4

Add CONTENT OK

The Calculator command "Add" adds the number you specify for CONTENT to the total of the accumulator you are using. The accumulated total is then ready for your next operation. You may also specify the Add command with a plus sign + . See also: arithmetic. 4A

Clear Accumulator/File... 4B

Accumulator: Clear Accumulator OK

The Calculator command "Clear Accumulator" sets the total of the accumulator you are using to zero. See also: accumulator. 4B1

File: Clear File OK

The Calculator command "Clear File" erases the contents of your calculator file except for the origin statement. See also: calculator-file. 4B2

Divide CONTENT OK:

The Calculator command "Divide" divides the value of the accumulator you are using by the number you specify for CONTENT. You may also specify the Divide command with the sign / . See also: arithmetic. 4C

Evaluate CONTENT OPERATOR OK

OPERATOR = Add, Subtract, Multiply, Divide, +, -, x, *, /

The Calculator command "Evaluate" performs the operation you specify for OPERATOR on the number you specify for CONTENT and the number in the accumulator you are using. The item you specify for CONTENT may be entered from the keyboard directly as a number or indirectly as a simple arithmetic expression (e.g. 1+9-2/4). You can skip the OPERATOR by typing <CA> when prompted by OK/C: Your CONTENT will then be added to the value of the accumulator you are using. If you specify an OPERATOR, the result will replace the value of the accumulator you are using. The number you specify for CONTENT will be followed by an asterisk when entered in your

Calculator file. 4D
Execute (universal command)
##<nls, execute>## 4E
Format...
  The Calculator command "Format" allows you to specify the format of numbers for the accumulator register you are using. Each accumulator register needs a separate format specification. You can indicate whether or not you want commas and/or dollar signs. Numbers can have up to 11 digits with from 0 to 5 places to the right of the decimal. The default format for numbers is right-justification, 2 places to the right of the decimal and 9 places to the left, no commas and no dollar signs. For example: 999999999.99. The Calculator Tool has internal checks that prevent you from entering numbers that do not fit your specified format, or choosing formats that cannot be used for numbers that already exist. Your Calculator file will show a format change with the first number after a Format command is given. 4F
  Commas: Format Commas ANSWER OK
    The Calculator command "Format Commas" places commas in your numbers like this: 999,999,999.99. Type "n" for ANSWER when you want to clear all of the commas from your accumulator. 4F1
  Dollar: Format Dollar (signs) ANSWER OK
    The Calculator command "Format Dollar" places dollar signs ($) in front of each number. Type "n" for ANSWER when you want to clear the $ from your accumulator. 4F2
  Left: Format Left (justify) OK
    The Calculator command "Format Left" writes numbers that appear with no spaces preceding them. See also: Format Right. 4F3
  Places: Format Places (to the) Right/Left CONTENT OK
    The Calculator command "Format Places" allows you to specify the number of digits that will print to the right or the left of the decimal point. The total number of printing digits allowed in a number is 11. Within this limit only 5 can follow the decimal. If you attempt to enter a number with more digits to the left of the decimal point than the current format specifies, the error message "Format too small for input" appears, the operation is not performed, and you must begin with another command. If you specify a format that is too small for the number in your accumulator, the error message "Format too small for accumulator, FORMAT RESET TO DEFAULT" appears, the operation is not performed, and the place format is changed back to the default. See also: format. 4F4
  Right: Format Right (justify) OK
    The Calculator command "Format Right" prints numbers flush right. If you have your decimal point all the way to the right (use the Format Places command), this will allow you space for two places in the billions. See also: Format Left. 4F5
Goto (universal command)
##<nls, goto>## 4G
Help (universal command)
##<nls, help>## 4H
Insert (accum following) STRING/STRUCTURE...
  The Calculator command "Insert" writes the value of the accumulator you are using into another file. 4I
  STRING: Insert (accum following) STRING DESTINATION OK

The Calculator command "Insert (accum following) STRING" writes the value of the accumulator you are using following the type of STRING you specify at the DESTINATION you specify. 4I1

STRUCTURE: Insert (accum following) STRUCTURE DESTINATION LEVEL-ADJUST OK

The Calculator command "Insert (accum following) STRUCTURE" inserts the value of the accumulator you are using following the type of STRUCTURE you specify at the DESTINATION you specify. 4I2

minus command: -

##<subtract>## 4J

Multiply CONTENT OK

The Calculator command "Multiply" multipies the value of the accumulator you are using by the number(s) you specify for CONTENT. You may also specify the Multiply command with the signs * or x.

See also: arithmetic. 4K

NUM: (adding at the herald)

a prompt that asks for a number. If you put in a number and follow this by an <OK>, it is added to the accumulator you are using. Any character that is not a number is simply ignored. 4L

plus command: +

##<add>## 4M

Quit

##<nls, quit>## 4N

<REPEAT> (<CTRL-B>) (universal command)

##<nls, repeat>## 4O

Replace STRING/STRUCTURE (at) DESTINATION (by accumulator) OK

The Calculator command "Replace" replaces a STRING or STRUCTURE (such as a visible or a branch) with the value of the accumulator you are using. 4P

semicolon command: ; (universal command)

##<nls, semicolon>## 4Q

Show... 4R

Accumulator: Show Accumulator (Registers) OK

The Calculator command "Show Accumulator" displays the values of the ten accumulators you are using.

File (DNLS only): Show File (in window) DESTINATION OK

The Calculator command "Show File" displays your calculator file in a window of your screen. While you have this file loaded, you can see the numbers you are entering into the accumulator you are using.

slash command: /

##<divide>## 4S

star command: *

##<multiply>## 4T

Subtract CONTENT OK

The Calculator command "Subtract" subtracts the number you specify for CONTENT from the number in the accumulator you are using. You may also specify the Subtract command with the minus sign - . See also: arithmetic 4U

Total OK

The Calculator command "Total" copies the number in the accumulator you are using to the end of your calculator file. 4V

Use...

The Calculator command "Use" allows you to specify the different accumulators you wish to use. 4W

Accumulator: Use Aacumulator (number) CONTENT OK

The Calculator command "Use Accumulator" allows you to specify for CONTENT the number of the accumulator you wish to use. You can specify any of the ten accumulators that are in the register of accumulators you are using. See also: Use Saved. 4W1

Saved: Use Saved (Accumulators) OK

The Calculator command "Use Saved" allows you to use the accumulator register you were using when you finished your previous session with the calculator. After giving this command, you will be using the accumulator from your last session that has the same number as the one you are presently using. 4W2

example of using saved accumulators

##<using>##

Write (new) File CONTENT OK:

The Calculator command "Write (new) File" creates a new file named whatever you specify for CONTENT, and moves your calculator file to it. (You cannot write on an already existing file.) The Calculator depends on the structure and format in the Calculator file. It is not possible to use the Calculator if this structure or format has been changed in any way. A new file may be edited in any way you desire. "Write File" also clears the calculator file of all entries. Therefore any subsequent "Write File" makes a new file containing only those entries since the last "Write File". 4X

% backlinks: <base, write>

x command

##<multiply>## 4Y

%

%Alphabetical List of the commands that are found only in the Calculator subsystem: (You still have available any of the universal commands.)

Add CONTENT OK 5A
Clear Accumulator OK 5B
Clear File OK 5C
Divide CONTENT OK 5D
Evaluate CONTENT OPERATOR OK 5E
Format Commas ANSWER OK 5F
Format Dollar (signs) ANSWER OK 5G
Format Left (justify) OK 5H
Format Places (to the) Right/Left CONTENT OK 5I
Format Right (justify) OK 5J
Insert (accum following) STRING DESTINATION OK 5K
Insert (accum following) STRUCTURE DESTINATION LEVEL-ADJUST OK 5L
Multiply CONTENT OK 5M
Replace STRING/STRUCTURE (at) DESTINATION (by accumulator) OK 5N
Show Accumulator (Registers) OK 5O
Show File (in window) DESTINATION OK 5P
Subtract CONTENT OK 5Q
Total OK 5R
Use Accumulator (number) CONTENT OK 5S
Use Saved (Accumulators) OK 5T
Write (new) File CONTENT OK 5U
You may also use any of the following arithmetic signs as commands 5V
    x command (multiply): X 5V1
    star command (multiply): * 5V2
    plus command (add): + 5V3
    minus command (subtract): - 5V4
    slash command (divide): / 5V5

% source-code for Calculator .VBS=0; .PBS; .IgRest; <:b> % 6

Maintenance-Index Branch
    DESCRIPTION
        Text descriptions of the tool or subsystem, describing the functions performed by the tool or subsystem.
    CHANGES
        DATE TIME IDENT <link to change>
            Text descriptions of changes made to the tool or subsystem, each change note should include the name of the person making the change, the date, and a reason for the change, as well as the description of the change.
    NEEDS AND POSSIBILITIES
        When CALC-IDENT file busy, causes infinite loop.
        Title command in TNLS caused infinite loop.
        Show File command doesn't split screen right.
    GRAMMAR SOURCE <links to the source branches of the grammar>
        Links to many files or branches may be placed in substructure.
         Library branch instructions may surround links.
        Naming Conventions
            File: <nine, toolname.nls,>
            Branch: <grammar>
    GRAMMAR COMPILED <links to the compiled files of the grammar>
        File Naming Convention: <nine, toolname.GRAM,>
    GRAMMAR COMPACTED <links to the compacted files of the grammar>
        File Naming Convention: <nine, toolname.CGR,>
    PARSE FUNCTIONS CODE SOURCE <links to the source branches of the

parse function code>
Naming Conventions
File: <nine, toolname-pfc.nls,>
Branch: <pfc>
PARSE FUNCTIONS CODE COMPILED <links to the compiled files of the parse function code>
File Naming Convention: <sysnine, toolname.PFC>
PARSE FUNCTIONS DATA SOURCE <links to the source branches of the parse function data>
Naming Conventions
File: <nine, toolname-pfd.nls,>
Branch: <pfd>
PARSE FUNCTIONS DATA COMPILED <links to the compiled files of the parse function code>
File Naming Convention: <sysnine, toolname.PFD>
BACKEND SOURCE <links to the source branches of the backend>
These should be divided into machine independent and operating system interface catagories. Links to many files or branches may be placed in substructure. Library branch instructions may surround links.
Naming Conventions
Branch: <be>
File: <directory, filefunction.NLS>
When ever more than one multi-file tool will occupy the same directory, the file naming convention is:
<directory, toolname-filefunction.NLS>
BACKEND COMPILED <link to the runable file of the backend>
For tools this is a SAV file (named .SAV), for subsystems this is a REL file (named .SUBSYS).
File Naming Conventions
<sysnine, toolname.SAV,>
<sysnine, toolname.SUBSYS,>
SHARED FILES <link to files shared with other tools>
USER DOCUMENTATION <links to the user documentation files>
There are several ways in which user documentation is provided, two being output printer files and user guides pointed to by help. TXT is used for sequential files that are not in output processor format.
Naming Conventions
Files:
<xhelp, toolname.nls,>
<nswdoc, toolname-userguide.nls,>
<nswdoc, toolname.txt,>
<nswdoc, toolname-userguide.txt,>
<nswdoc, toolname.print,>
<nswdoc, toolname-userguide.print,>
Branches:
<toolname>
<userguide>
MAINTENANCE DOCUMENT <links to documents describing the internal procedures and data structures of this tool or subsystem>
This documentation is intended to be useful to persons debugging or upgrading the tool or subsystem.
Naming Conventions
File: <docnine, toolname-MAINTDOC.NLS,>
Branch: <MAINTDOC>

```
   DATASTRUCTURE DOCUMENT
      links to documents describing the interfaces to this tool or
      subsystem, especially any data structures shared with other
      tools or subsystems.
         Naming Conventions
            File:  <docnine, toolname-DATASTRUCDOC.NLS,>
            Branch:  <DATASTRUCDOC>
   RUNFILES SOURCE <links to NLS RUNFILE source branches>
      that are useful in building new versions of this tool or
      subsystem
      Naming Conventions
         File:  <docnine, toolname-RUNFILES.NLS,>
         Branch:  <RUNFILES>
   RUNFILES <links to RUNFILES>
      File Naming Convention:  <relnine, toolname.RUN,>
% grammar source code branch %                                    6B
   FILE csubsysname  % (arcsubsys, cml10,)  (relnine,
   calculator.cgr,) %
   %COMPILE INSTRUCTIONS%                                        6B2
      INCLUDE <nine, nls-grammar, flags !subsystems>
   %DECLARATIONS%                                                6B3
      INCLUDE <nine, nls-grammar, declarations !universal>       6B3A
      DECLARE FUNCTION                                           6B3B
         xcshoaccums, xcshofil, xclraccum, xclrfil, xceval, xcevend,
         xfdigits, xcomma, xcjust, xcfeedb, xdollar, xcinsert,
         xcarith, xcreplace, xctotl, xcuseaccum, xcusesaved,
         xcwritef, xcreenter, xcquit, xcalcinit;                 6B3B1
      DECLARE COMMAND WORD %subsystem command words (should be 100
      to 127)%                                                   6B3C
         "ADD" = 100 ,
         "SUBTRACT" = 101 ,
         "MULTIPLY" = 102 ,
         "DIVIDE" = 103 ,
         "RIGHT" = 104 ,
         "LEFT" = 105 ;
   %COMMON RULES%                                                6B4
      INCLUDE <nine, nls-grammar, rules !universal>
   %COMMANDS% SUBSYSTEM calculator KEYWORD "CALCULATOR"          6B5
      INITIALIZATION                                             6B5A
         zcinit= % order switched to accomodate compiler-compacter
         problem %                                               6B5A1
            ( ( IF NOT DISPLAY IF RESULT SHOW(RESULT) )
            / IF NOT xcalcinit() ( feendsubsys() xquitsubsys() ) );
      TERMINATION                                                6B5B
         zcterm=                                                 6B5B1
             xcquit();
      RENTRY                                                     6B5C
         zclcrent = xcreenter()                                  6B5C1
             IF NOT RESULT ( feendsubsys() xquitsubsys() );

      add COMMAND = ("ADD")                                      6B5D
         param2 _ LSEL(#"NUMBER")                                6B5D1
         CONFIRM                                                 6B5D2
         %  execute ADD (param2)    %
          xcarith(#"ADD",param2);
      clear COMMAND = "CLEAR"                                    6B5E
```

```
      ("ACCUMULATOR"
         CONFIRM                                                  6B5E1A
          xclraccum()
      /"FILE"
         CONFIRM                                                  6B5E2A
         xclrfil()                                                6B5E2B
      );
   divide COMMAND = ("DIVIDE")                                     6B5F
      param2 _ LSEL(#"NUMBER")                                     6B5F1
      CONFIRM                                                      6B5F2
      %  execute DIVIDE (param2)   %
       xcarith(#"DIVIDE",param2);

   evaluate COMMAND = "EVALUATE"!L2!                               6B5G
      param _ LSEL(#"TEXT")                                        6B5G1
       xceval(param)
       SHOW(RESULT)
      (("ADD"  / "+"/ CONFIRM )
         param _ #"ADD"                                            6B5G4A
      /("DIVIDE"  / "/")
         param _ #"DIVIDE"                                         6B5G5A
         CONFIRM                                                   6B5G5B
      /("MULTIPLY"  / "*"  / "X")
         param _ #"MULTIPLY"                                       6B5G6A
         CONFIRM                                                   6B5G6B
      /("SUBTRACT"   / "-")
         param _ #"SUBTRACT"                                       6B5G7A
         CONFIRM                                                   6B5G7B
      )
       xcevend(param);
   format COMMAND = "FORMAT"                                       6B5H
      ("PLACES" <"to the">
         param _                                                   6B5H1A
         ("RIGHT"
         /"LEFT"
         )
         <"of decimal point">
         param2 _ LSEL(#"NUMBER")                                  6B5H1F
            CONFIRM                                                6B5H1F1
             xfdigits(param,param2)
      /"COMMAS"
         param _ ANSWER                                            6B5H2A
         CONFIRM                                                   6B5H2B
          xcomma(param)
      /param _
         ("LEFT"
         /"RIGHT")
         <"justify">
         CONFIRM                                                   6B5H3D
          xcjust(param)
      / IF NOT DISPLAY "TERSE" <"output">
         param _ ANSWER                                            6B5H4A
         CONFIRM                                                   6B5H4B
         xcfeedb(param)                                            6B5H4C
      /"DOLLAR" <"signs">
         param _ ANSWER                                            6B5H5A
```

```
      CONFIRM                                                    6B5H5B
      xdollar(param)                                             6B5H5C
   ) ;
insert COMMAND = "INSERT"                                        6B5I
   <"accum following">
   level _ NULL                                                  6B5I2
   (ent _ ( text1 / "LINK" / "NUMBER" ) <"at">
      dest _ DSEL(ent)                                           6B5I3A
   / ent _ "TEXT" <"at">
      dest _ DSEL(#"CHARACTER")                                  6B5I4A
   / ent _ structure <"at">
      dest _ DSEL(#"STATEMENT")                                  6B5I5A
      level _ levadj())                                          6B5I5B
   CONFIRM                                                       6B5I6
   %  execute INSERT ACCUM (dest, param)    %
    xcinsert(ent, dest, level);
multiply COMMAND = ("MULTIPLY")                                  6B5J
   param2 _ LSEL(#"NUMBER")                                      6B5J1
   CONFIRM                                                       6B5J2
   %  execute cmult (param2)   %
    xcarith(#"MULTIPLY",param2) ;
number COMMAND =  param2 _ LSEL(#"NUMBER")                       6B5K
   CONFIRM                                                       6B5K1
    xcarith(#"ADD",param2) ;

replace COMMAND = "REPLACE"                                      6B5L
   dent _ editentity <"at">                                      6B5L1
   dest _ DSEL (dent)                                            6B5L2
   sent _ #"TEXT"                                               6B5L3
   <"by accumulator">
   CONFIRM                                                       6B5L5
   %  execute replace    %
   xcreplace(dent,dest,sent);                                    6B5L7
show COMMAND = "SHOW"!L2!                                        6B5M
   ("ACCUMULATOR"  <"Registers">
      CONFIRM                                                    6B5M1A
      xcshoaccums()                                              6B5M1B
      IF RESULT                                                  6B5M1C
         ( IF DISPLAY SHOWCONFIRM(RESULT)
         / IF NOT DISPLAY SHOW(RESULT) )
   /IF DISPLAY "FILE" <"in window">
      DSEL(#"WINDOW")                                            6B5M2A
      CONFIRM                                                    6B5M2B
       xcshofil(WINDOW)
   );
subtract COMMAND = ("SUBTRACT")                                  6B5N
   param2 _ LSEL(#"NUMBER")                                      6B5N1
   CONFIRM                                                       6B5N2
   %  execute SUBTRACT (param2)   %
    xcarith(#"SUBTRACT",param2) ;
total COMMAND = "TOTAL"                                          6B5O
   CONFIRM                                                       6B5O1
    xctotl()
   ;
use COMMAND = "USE"                                              6B5P
   ("ACCUMULATOR"  <"number">
```

```
            param2 _ LSEL(#"NUMBER")                                    6B5P1A
            CONFIRM                                                     6B5P1B
             xcuseaccum(param2)
          /"SAVED"  <"Accumulators">
            CONFIRM                                                     6B5P2A
             xcusesaved()
          ) ;
       write COMMAND = "WRITE" <"copy of calculator file named">        6B5Q
          param _ LSEL(#"NEWFILENAME")                                  6B5Q1
          CONFIRM                                                       6B5Q2
          IF NOT xcWritef(param) ( xquitsubsys() feendsubsys() )
                                                                        6B5Q3
          ;
       INCLUDE <nine, nls-grammar, commands !universal>                 6B5R
       END.                                                             6B5S
   FINISH
```

```
% be source code branch %                                              6C
   FILE pscalc % (arcsubsys, 1109,) to  (relnine,
   calculator.subsys,) %
   % DECLARATIONS %
      REF cda, coldda;
      %CALCULATOR ERROR MESSAGES%
         (spliterr) STRING = "Need a larger window";                  6C2B1
         (formdigerr) STRING = "too many digits-default format set";
                                                                      6C2B2
         (badaccno) STRING = "Use a value between 1 and 10";          6C2B3
         (caupderr) STRING = "System error: Unable to re-open
          calc-ident file";                                           6C2B4
         (calsyserr) STRING = "CALCULATOR SYSTEM ERROR";              6C2B5
         (acsaverr) STRING = "No saved accumulators found";           6C2B6
         (badcfile) STRING = "Bad Calc-Ident file; unable to go on";
                                                                      6C2B7
         (blanks) STRING = "                    ";
                                                                      6C2B8
      (calculator) EXTERNAL _ (                                        6C2C
         $"XCALCINIT", $xcalcinit,
         $"XCARITH", $xcarith,
         $"XCSHOACCUMS", $xcshoaccums,
         $"XCSHOFIL", $xcshofil,
         $"XCLRACCUM", $xclraccum,
         $"XCLRFIL", $xclrfil,
         $"XCEVAL", $xceval,
         $"XCEVEND", $xcevend,
         $"XFDIGITS", $xfdigits,
         $"XCOMMA", $xcomma,
         $"XCJUST", $xcjust,
         $"XCFEEDB", $xcfeedb,
         $"XDOLLAR", $xdollar,
         $"XCINSERT", $xcinsert,
         $"XCREPLACE", $xcreplace,
         $"XCTOTL", $xctotl,
         $"XCUSEACCUM", $xcuseaccum,
         $"XCUSESAVED", $xcusesaved,
         $"XCWRITEF", $xcwritef,
         $"XCREENTER", $xcreenter,
         $"XCQUIT", $xcquit,
         0,0);
   % CALCULATOR SUBSYSTEM %
      % INITIALIZATION %
         (xcalcinit) PROCEDURE (rtnlist REF);                         6C3A1
            LOCAL
               fileno;   %file number of calc-ident%
            LOCAL TEXT POINTER tp1, tp2;
            LOCAL STRING flnam[50];
            % load and set up calculator file %
               IF NOT cafilinitialize() THEN RETURN; % null retnlist
               is interpreted as a FALSE return %
            % clear all accumulators %
               caclear(0, 10);
            % initialize %
               asub _ 0;
```

```
            *opstring* _ NULL;
            *signstr* _ "+;
            *astrng* _ "1; %default is first accumulator%
            %mark beginning of this calculator entry in file%
               tp1 _ tp2 _ $"******************";
               tp1.stastr _ 1;
               tp2.stastr _ 1;
               tp1[1] _ 1;
               tp2[1] _ 18;
               castid _ cinssta(castid,sucdir,$tp1,$tp2);
            proc _ $qcadd;
            qfloutp($accum, $acstring,2);%convert starting accum%
            #rtnlist#[1] _ TRUE; % setup default returnlist %
            CASE nlmode OF = typewriter: #rtnlist#[1] _ USE
            rtnstring($acstring);
            ENDCASE
               BEGIN
               dismes(1,$acstring); %display starting accums%
               dpset(dspno, endfil, endfil, endfil);
               END;
         % exit back to parser/control %
            RETURN(TRUE);
         END.

% FEEDBACK, COMMAND INITIALIZATION %
   (cfeedback) PROCEDURE; % do calculator feedback %        6C3B1
      LOCAL STRING mess[35];
      % floating result to string %
         qfloutp($accum+asub, $acstring,2);
      % see if special formatting char, $ %
         IF cadflg % global in DATA % THEN *acstring* _
         "$,*acstring*;
         litapflag _ FALSE; %to prevent bypassing rstlit in
         setlit%
      CASE nlmode OF
         = fulldisplay:
            BEGIN
            % indicate which accum being used and its
            contents%
               *mess* _ "accumulator  #";
               *mess* _ *mess*, *astrng*, ":  ", *acstring*;
               dismes(1, $mess);
            END;
      ENDCASE
         BEGIN
         % old TNLS bottom of loop %
             %handle TNL hard copy feedback%
               IF NOT nofeedbk THEN % long form %
                  BEGIN
                  *lit* _ *blanks*, *opstring*, "    ",
                  *acstring*;
                  dismes(1,$lit);
                  END;
         END;
      %set defaults %
         proc _ $qcadd;
```

```
            *opstring* _ NULL;
            *signstr* _ '+;
            namereset _ TRUE;
         RETURN;
         END.

% ARITHMETIC %
   (xcarith) % execute arithmatic %
   PROCEDURE  (operation REF, numptr REF);                    6C3C1
      % declarations %
         LOCAL optype, acind;
         LOCAL ptr1 REF, ptr2 REF;
         LOCAL STRING char[1], locstr[100];
      % get pointers %
         &ptr1 _ ELEM#numptr#[tppair];
         &ptr2 _ &ptr1+d2sel;
      % determine optype for ADD %
         CCPOS ptr1;
         IF ELEM#operation#[cwtype] = 100 % add % THEN
            CASE TRUE OF % set operator type %
               = (FIND ('+ / 'a) ^ptr1): optype _ 100;
               = (FIND ('- / 's) ^ptr1): optype _ 101;
               = (FIND ('*/'x/'m) ^ptr1): optype _ 102;
               = (FIND ('/ / 'd) ^ptr1): optype _ 103;
               ENDCASE optype _ 100
         ELSE optype _ ELEM#operation#[cwtype] ;
      ceval(&ptr1, &ptr2);  % evaluate expression %
      opconvert(optype); % set up proc global to be called by
      cevend %
      cevend(); % use proc global on accumulator %
      RETURN;
      END.

   (ceval) % evaluate %
   PROCEDURE (ptr1 REF, ptr3 REF) ;                           6C3C2
      % Given pointers to the beginning and ending of a string
      containing an arithmetic expression, updates vaccum and
      opstring to contain correct value %
      % DECLARATIONS %
         LOCAL
            param REF,
            char,      %temp for parsing%
            acflag;    %true when accum value is input to
            'value' expression%
         LOCAL TEXT POINTER tptr, ptr2;
      acflag _ FALSE;
      vaccum[0] _ vaccum[1] _ 0;
      LOOP
         BEGIN
         FIND ptr1 >;
         CASE TRUE OF
            = (FIND ('+/'a/SP) ^ptr1): proc _ $qcadd;
            = (FIND ('-/'s) ^ptr1): proc _ $qcsub;
            = (FIND ('*/'x/'m) ^ptr1): proc _ $qcmult;
            = (FIND ('//'d) ^ptr1): proc _ $qcdiv;
            = (FIND (D/'.) ): proc _ $qcadd;
```

```
            = (FIND '# ^ptr1): acflag _ TRUE;
            ENDCASE err($"expression invalid");
         LOOP
            BEGIN
            IF acflag THEN
               BEGIN
               IF NOT FIND ptr1 > 1$2D ^ptr2
               THEN err($"invalid expression");
               *tstrng* _ ptr1 ptr2;
               char _ VALUE($tstrng)*2-2;
               [proc]($vaccum,$accum+char);
               acflag _ FALSE;
               EXIT LOOP;
               END
            ELSE
               BEGIN
               IF FIND ptr1 > '# ^ptr1 THEN
                  BEGIN
                  acflag _ TRUE;
                  REPEAT LOOP;
                  END;
               FIND ptr1 > ('+/'-/TRUE) $(D/'./',) ^ptr2;
               *opstring* _ ptr1 ptr2;
               FIND SF(*opstring*) ^tptr;
               IF nfloat($opstring,$opfloat, $opfloat + 1)
               THEN
               qcneg($opfloat); %convert to floating point and
               take care of sign %
               [proc]($vaccum,$opfloat);
               EXIT LOOP;
               END;
            END;
         ptr1 _ ptr2;
         ptr1[1] _ ptr2[1];
         IF ptr2[1] >= ptr3[1] THEN EXIT LOOP;
         END;
      RETURN;
      END.

(cevend) %operate on accum use temp accum, vaccum, as
operand%
PROCEDURE;                                                    6C3C3
   qcins($vaccum, $opstring); %insert value of expression
   in CALC file%
   [proc]($accum+asub,$vaccum);
   cfeedback();  % show it to the user %
   RETURN;
   END.

(opconvert)  % convert commandword number to procedure name
%
PROCEDURE (optype);                                           6C3C4
   CASE optype OF
      = 100 %- add -%:
         BEGIN
         *signstr* _ '+;
```

```
                proc _ $qcadd;
                END;
             = 101 %- subtract -%:
                BEGIN
                *signstr* _ "-;
                proc _ $qcsub;
                END;
             = 102 %- multiply -%:
                BEGIN
                *signstr* _ "*;
                proc _ $qcmult;
                END;
             = 103 %- divide -%:
                BEGIN
                *signstr* _ "/;
                proc _ $qcdiv;
                END;
             ENDCASE err($"commandword number mismatch in
             opconvert");
          RETURN;
          END.

      % CLEAR ACCUMULATOR(S) %
       (xclraccum) PROCEDURE;                                    6C3D1
          caclear(asub, 1);
          CASE nlmode OF
             = fulldisplay: dpset(dspno, endfil, endfil, endfil);
          ENDCASE;
          cfeedback();  % show it to the user %
          RETURN;
          END.

      % CLEAR FILE %
       (xclrfil) PROCEDURE;                                      6C3E1
          resetf(castid.stfile); % reset pc %
          castid.stpsid _ cda.dacsp.stpsid _ orgstid;
          CASE cda.daauxiliary OF
             = TRUE: % not showing file %
                dpset(dspno, endfil, endfil, endfil);
             ENDCASE
                dpset(dspyes, castid, endfil, endfil);
          RETURN;
          END.

      % EVALUATE EXPRESSION %
       (xceval) %Calculate value of an expression%
       PROCEDURE (paramptr REF, rtnlist REF);                    6C3F1
          ceval(ELEM#paramptr#[tppair],
          ELEM#paramptr#[tppair]+d2sel);
          qfloutp($vaccum,$opstring,2);
          #rtnlist#[1] _ USE rtnstring($opstring);
          RETURN;
          END.

       (xcevend) % execute end of calculator evaluate command%
       PROCEDURE (operation REF);                                6C3F2
```

```
            opconvert(ELEM#operation#[cwtype]);
            cevend();
            RETURN;
            END.

         % FORMAT CHANGE %
            (xfdigits) %set number of digits after the decimal%
            PROCEDURE (param REF %LEFT of RIGHT of decimanal %, value
            %number of digits %);                                    6C3G1
               LOCAL cacadflg, cafld1,char;
               LOCAL STRING cafstr[20];
               getpstring(value, $cafstr);
               char _ VALUE($cafstr);
               CASE ELEM#param#[cwtype] OF
                  = 104 %- right -%:
                     BEGIN
                     IF char NOT IN [0,5] THEN
                        BEGIN
                        err($formdigerr);
                        END
                     ELSE
                        BEGIN
                        dfoutm.fld2 _ char;
                        dfoutm.fld1 _ 12 - char;
                        END;
                     END;
                  = 105 %- left -%:
                     BEGIN
                     IF char NOT IN [0,9] THEN
                        BEGIN
                        err($formdigerr);
                        END
                     ELSE
                        BEGIN
                        dfoutm.fld1 _ char;
                        dfoutm.round _ dfoutm.fld1 + dfoutm.fld2;
                        IF dfoutm.round > 12 THEN
                           BEGIN
                           dfoutm _ 064014120200B;
                           err($formdigerr);
                           RETURN;
                           END;
                        END;
                     END;
               ENDCASE NULL;
               cfeedback();  % show it to the user %
               RETURN;
               END.
            (xcjust) %right or left justify number %
            PROCEDURE (param REF);   %LEFT or RIGHT of decimal %   6C3G2
               CASE ELEM#param#[cwtype] OF
                  = 104 %- right -%:
                     BEGIN
                     dfoutm.just _ 1;
                     calflg _ FALSE;
                     END;
```

```
            = 105 %- left -%:
               BEGIN
               IF cacflg THEN
                  BEGIN
                  calflg _ TRUE;
                  dfoutm.just _ 1;
                  END
               ELSE
                  BEGIN
                  dfoutm.just _ 3;
                  calflg _ FALSE;
                  END;
               END;
         ENDCASE;
         cfeedback();  % show it to the user %
         RETURN;
         END.
      (xcomma) % set flag to insert commas in formatted number%
      PROCEDURE (param REF);                                    6C3G3
         IF param = 1 THEN
            cacflg _ TRUE
         ELSE cacflg _ FALSE;
         cfeedback();  % show it to the user %
         RETURN;
         END.
      (xdollar) % set flag to insert dollar sign in formatted
      number%
      PROCEDURE (param REF);                                    6C3G4
         IF param = 1 THEN
            cadflg _ TRUE
         ELSE cadflg _ FALSE;
         cfeedback();  % show it to the user %
         RETURN;
         END.
      (xcfeedb) % set flag to abbreviate feedback in TNLS%
      PROCEDURE (param REF);                                    6C3G5
         IF param = 1 THEN
            nofeedbk _ TRUE
         ELSE nofeedbk _ FALSE;
         cfeedback();  % show it to the user %
         RETURN;
         END.
   % SHOW ACCUMULATORS %
      (xcshoaccums) % show accumulators % PROCEDURE (rtnlist
      REF);                                                     6C3H1
         %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%
         LOCAL index;
         LOCAL STRING acstr[30], ord[5], destring[350];

         index _ 0;
         *destring* _ NULL;
```

```
        DO
           BEGIN
           gfloutp($accum + index, $acstr, 2); %convert%
           *ord* _ STRING((index+2)/2), ": ";
           *destring* _ *destring*, EOL, "  ", *ord*, *acstr*;
           END
        UNTIL (index _ index + 2) > 19;
        dpset(dspno, endfil, endfil, endfil);
        #rtnlist#[1] _ USE rtnstring($destring);
        RETURN;
        END.

% SHOW FILE %
   (xcshofil) PROCEDURE;                                          6C3I1
      LOCAL lnewda REF, da REF, csp;
      LOCAL LIST atbug[3], dspinbug[3];
      &da _ lda(); % address of da being split %
      % set up lists %
         #atbug# _ da.daleft+23, da.datop, da.dawid;
         #dspinbug# _ da.daright, da.datop, da.dawid;
      % split screen %
         &coldda _ wbreak($atbug, 0, vertical, $dspinbug:
         &lnewda);
      % get rid of old calculator da %
         curmkr _ csp _ cda.dacsp;
         delda(&cda);
         cspupdate _ &cda _ &lnewda;
         cda.dacsp _ csp;
      dpset(dspallf, cda.dacsp, endfile, endfile);
      RETURN;
      END.


% TOTAL %
   (xctotl) PROCEDURE;                                            6C3J1
      *signstr* _ "T;
      qcins($accum + asub, $opstring);
      dpset(dspno, endfil, endfil, endfil);
      CASE nlmode OF
         = typewriter: dismes(1, $opstring);
      ENDCASE;
      *signstr* _ "+;
      RETURN;
      END.

% USE ACCUMULATOR # %
   (xcuseaccum) PROCEDURE(numptr);                                6C3K1
      LOCAL index;
      getpstring(numptr, $astrng);
      IF (index _ VALUE($astrng) * 2 - 2) NOT IN [0,19] THEN
         err($badaccno)
      ELSE
         % update the master window !! %
         asub _ index; % change main subscript %
      dpset(dspno, endfil, endfil, endfil);
      cfeedback();  % show it to the user %
```

```
        RETURN;
        END.

    % USE SAVED ACCUMULATORS %
      (xcusesaved) PROCEDURE (rtnlist REF);                    6C3L1
        LOCAL stid, index;
        LOCAL STRING
          temstr[4],
          calcstr[4],    % special calc signature %
          sigstr[35],   % for checking signature %
          value[22];
        LOCAL TEXT POINTER start, end;
        %initialize locals%
          *calcstr* _ "CALC";
          index _ 0;
          stid _ orgstid;
          stid.stfile _ cda.dacsp.stfile;
          stid _ <FILMNP, getsub>(stid); %location of info%
          fechsig(stid, $sigstr);
          *temstr* _ *sigstr*[1 TO 4];
          CCPOS SF(stid);
          IF *temstr* # *calcstr* THEN err($acsaverr);
        %check for nothing saved%
          IF NOT FIND ["CALC ACCUMS:"] ^start THEN
            BEGIN
            IF NOT FIND start ["/] <CH ^end THEN
            err($acsaverr);
            END;
          %get and convert values%
          DO
            BEGIN
            IF NOT FIND start ["/] <CH ^end THEN
            err($acsaverr);
            *value* _ start end;
            IF nfloat($value, $accum + index, $accum + index +
            1) THEN qcneg($accum+index);
            FIND end >CH ^start; %get over slash delimiter%
            END
          UNTIL (index _ index + 2) > 19;
        %retrieve format variables%
          stid _ <FILMNP, getsuc>(stid); %location of format
          flags%
          *sigstr* _ NULL;
          fechsig(stid, $sigstr);
          *temstr* _ *sigstr*[1 TO 4];
          IF *temstr* # *calcstr* THEN err($acsaverr);
          CCPOS SF(stid);
          IF NOT FIND ["FORMAT:"] ^start THEN err($acsaverr);
          IF NOT FIND  3D ^end THEN err($acsaverr);
          *value* _ start end;
          cacflg _ IF *value*[1] = "0 THEN FALSE ELSE TRUE;
          cadflg _ IF *value*[2] = "0 THEN FALSE ELSE TRUE;
          calflg _ IF *value*[3] = "0 THEN FALSE ELSE TRUE;
        %retrieve format mask%
          stid _ getsuc(stid); %location of format mask%
          *sigstr* _ NULL;
```

```
            fechsig(stid, $sigstr);
            *temstr* _ *sigstr*[1 TO 4];
            IF *temstr* # *calcstr* THEN err($acsaverr);
            CCPOS SF(stid);
            FIND SF(stid) ^start;
            IF NOT FIND 1$12D ^end THEN err($acsaverr);
            *value* _ start end;
            dfoutm _ VALUE($value);
            %code to verify format may be inserted here - if user
            has messed with this statement, he may get a sudden
            illegal instruction eventually%
         dpset(dspno, endfil, endfil, endfil);
         cfeedback();  % show it to the user %
         RETURN;
         END.


% WRITE NEW FILE %
   (xcwritef) PROCEDURE (fnamptr REF); %calculator write file%
                                                          6C3M1
      %allow user to update the calc-ident file to a new file
      in his directory.%
      LOCAL STRING oldnam[50], relfilname[200];
      LOCAL nameaddress;
      % move file name to local string %
         CASE lnbfls( &fnamptr, 0, $relfilname) OF
            = lhostn: NULL;
            ENDCASE
               err($"Remote File Manipulations Not Implemented
               Yet");
      nameaddress _ fnamptr.RH;
      %do the actual update function%
         updtfl(castid.stfile, newversion, $relfilname);
         dismes(2, [flntadr(castid.stfile)].flastr);
            %calling routine will get frefint to close the
            file updated to%
      %re-open the old base file%
         clcname($oldnam); %set calc-ident file name%
         castid _ orgstid;
         IF NOT (castid.stfile _ <CORENL,
         cloafil>($oldnam))THEN
            BEGIN
            dismes(1, $caupderr);
            RETURN(FALSE);
            END;
         cda.dacsp _ castid;
      dpset(dspno, endfil, endfil, endfil);
      RETURN(TRUE);
      END.

% INSERT %
   (xcinsert) %Insert %
   PROCEDURE (ent, dest, level) ;                         6C3N1
      LOCAL LIST adrexp[3];
      LOCAL block[4];
      % set up global string to fool xinsert %
```

```
            block _ block[d2sel] _ $acstring;
            block[d2sel].stastr _ block.stastr _1;
            block[1] _ 1;
            block[d2sel+1] _ acstring.L + 1;
            #adrexp# _
               USE makedesc(uindex, ELEM#ent#[cwtype], FALSE),
               USE makedesc(utpblo, $block, FALSE),
               cwndow;
         xinsert(ent, dest, level, $adrexp);
         RETURN;
         END.

% REPLACE %
   (xcreplace) %replace accum %
   PROCEDURE (ent REF, dest, level) ;                          6C301
      LOCAL LIST adrexp[3];
      LOCAL block[4];
      % set up global string to fool xreplace %
         block _ block[d2sel] _ $acstring;
         block[d2sel].stastr _ block.stastr _1;
         block[1] _ 1;
         block[d2sel+1] _ acstring.L + 1;
         #adrexp# _
            USE makedesc(uindex, ELEM#ent#[cwtype], FALSE),
            USE makedesc(utpblo, $block, FALSE),
            cwndow;
      xreplace(dest, $adrexp, 0, 0);
      RETURN;
      END.

% RE-ENTER SUBSYSTEM CODE %
   (xcreenter) % calc reenter %
   PROCEDURE (rtnlist REF);                                    6C3P1
      LOCAL TEXT POINTER
         tp1, tp2, h1, h2, u1, u2, f1, f2, n1, n2, v1, v2;
      LOCAL STRING
         oldname[30], % calc-ident string only %
         nowname[70]; % whole name, currently in cda %
      #rtnlist#[1] _ TRUE;
      IF cda.daexis % our global still a da %
      AND NOT cda.daauxiliary % being viewed %
      THEN
         BEGIN
         clcname($oldname);
         filnam(cda.dacsp.stfile, $nowname);
         FIND SF(*nowname*) ^tp1;
         lnbfls( $tp1, 0, $nowname);
         IF NOT FIND tp1 > [".] < CH ^f2 THEN
            err($"system string error");
         *nowname* _ tp1 f2;
         *nowname* _ *nowname*[1 TO oldname.L];
         IF ( *nowname* # *oldname* ) % he loaded another file
         there%
            AND NOT cafilinitialize() THEN #rtnlist#[1] _
            FALSE % punt %
         ELSE
```

```
            BEGIN % good da, right file %
               getail(castid); % he may have added something - we
               were in this plex before %
               dpset(dspyes, castid, endfil, endfil);%recreate%
               END;
            END
         ELSE IF NOT cda.daexis
            AND NOT cafilinitialize() THEN #rtnlist#[1] _ FALSE;
            % punt %
         RETURN;
         END.


      % TERMINATION CODE %
         (xcquit) PROCEDURE; % calculator TERMINATION CODE %     6C3Q1
            % DECLARATIONS %
               LOCAL stid,   %cal-file origin%
                  da REF,
                  account,     % number of words of information to
                  save %
                  savsig,  % temp for NLS signature value %
                  trapping,    % TRUE if we have disarmed control C
                  %
                  capsav,    % save capabilities while trapping so
                  they can be restored when contol c is rearmed %
                  index;     %accumulator save array index%
               LOCAL STRING
                  save [250] , %enough for 10 accums%
                  errsr[150], % for error message from coropnfil %
                  calcsig[5],
                  temp[20];
               LOCAL TEXT POINTER start, finish;
               LOCAL LIST frombug[3], tobug[3];
            trapping _ FALSE;
            namereset _ FALSE;
            cbadent _ FALSE;
            % save state %
            *calcsig* _ "CALC";
            savsig _ cinit; % standard NLS signature %
            INVOKE(cinitsig);
            trapping _ TRUE;
            capsav _ trapcc();
            cinit _ setcinit($calcsig);
            account _ 19; %number of words - 1%
            *save* _ "CALC ACCUMS:";
            index _ 0;
            %convert accumulators to character string%
               DO
                  BEGIN
                  gfloutp($accum + index, $temp,2);
                  *save* _ *save*, *temp*, "/ ;
                  END
               UNTIL (index _ index + 2) > account;
               *save* _ *save*, "; ;
               stid _ cda.dacsp;
               stid.stpsid _ orgstid;
```

```
            %store accumulators in user file%
               start _ finish _ $save;
               start.stastr _ 1;
               finish.stastr _ 1;
               start[1] _ 1;
               finish[1] _ save.L + 1;
               stid _ cinssta(stid, sucdir, $start, $finish);
               *save* _ "FORMAT:";
               *save* _ *save*, STRING(cacflg);
               *save* _ *save*, STRING(cadflg);
               *save* _ *save*, STRING(calflg);
            %store format information in user file%
               finish[1] _ save.L + 1;
               stid _ cinssta(stid, sucdir, $start, $finish);
               *save* _ STRING(dfoutm);
            %store format mask in user file%
               finish[1] _ save.L + 1;
               stid _ cinssta(stid, sucdir, $start, $finish);
            cinit _ savsig;
            IF trapping := FALSE THEN notrapcc(capsav);
            CASE cda.daauxiliary OF
               = TRUE: % TNLS or DNLS with no displayed file %
                  delda(&cda); % file will get closed with next
                  recreate display %
               ENDCASE
                  BEGIN % set up lists %
                  #frombug# _ cda.daleft, cda.datop, cda.dawid;
                  #tobug# _ coldda.daleft, coldda.datop,
                  coldda.dawid;
                  &da _ wappend($frombug, $tobug); % call delete
                  window %
                  dafrmt(&da,0);
                  END;
            RETURN;
            (cinitsig) CATCHPHRASE;                                  6C3Q1W
               BEGIN
               CASE SIGNALTYPE OF = aborttype:
                  BEGIN
                  % reset ident string %
                     cinit _ savsig;
                  % re-arm control C if necessary. %
                     IF trapping := FALSE THEN notrapcc(capsav);
                  END;
               ENDCASE;
               CONTINUE;
               END;
            END.

      % INITIALIZATION %
         (cafilinitialize) PROCEDURE; % calc file set up %           6C4A
            LOCAL fileno;
            LOCAL STRING flnam[50];
            dpset(dspno,endfil,endfil,endfil); % set globals = no
            recreate %
            % get auxiliary da %
               &cda _ newda(); %get new da%
```

```
            intdafl(&cda); %initialize newda%
            cda.daauxiliary _ TRUE; % use this flag to tell whether
            to bother with screen %
         %load or initialize CALC-IDENT file%
            cda.dacsp _ orgstid;
            IF NOT cacfile (:fileno) THEN
               BEGIN
               dismes(1, $badcfile);
               RETURN(FALSE);
               END;
            cda.dacsp.stfile _ fileno;
         %initialize da to display file%
            castid _ cda.dacsp _ <STRMNP, getail> (<FILMNP, getsub>
               (cda.dacsp));
            IF cda.dacsp.stpsid = origin %empty file% THEN
               BEGIN
               clcname($flnam);
               updtfl(fileno, FALSE, $flnam); %don't care if update
               actually happened or not%
               END;
         RETURN (TRUE);
         END.

% CLEAR ACCUMULATOR(S) %
   (caclear) PROCEDURE (accx,clrwhat); %clear accum pointed at by
   accx or all accums%                                          6C5A
      IF clrwhat = 1 THEN
         BEGIN
         accum[accx] _ accum[accx+1] _ 0;
         RETURN;
         END
      ELSE
         BEGIN
         accx _ 0;
         DO
            accum[accx] _ accum[accx+1] _ 0
         UNTIL (accx _ accx +2) >= 19;
         END;
      RETURN;
      END.

FINISH of PSCALC
```