SYSGD Binders (5)

5 <NINE> SYSGD — 120 pgs

1

ADRMNP
AUXCOD
BACKVIEW
BCONST
BDATA ← BEIGNORE
BESYMMNP
BINTNLS
BRECORDS
CATNUM
CCALC

2

COLSRT
CORESUPPORT
DECIMITERS
DPSMIDDLE
DSPGEN
ENCAPSULATOR
ERROR
EXECFL
FILMNP
FRONTEND

3

IDENTSUPPORT
IOEXEC
INTERFACE-SHAREDPAGE
INTERFACE-NSW-DIRECT (1 proc, 1 pg. INSWD)
ISP
(x) LIO DATA
(x) LIO LRP
(x) LIO RUNTIME
MACONST
MA DATA

(CONT'd)

3 (contd)

MCONST
MDATAF
MIDDLE
MSHARED
MSG3DATA
MSG3FORMAT
NLSACNV
NLSGRAM
NLSPARSE

4

OPTAB
PCPB8-10
PSEDT1
PSEDT2
PSEXEC
PSHELP
PSPROGRAMS

5

RECFIL
SDATA
SEMANTICHELP
SEQFIL
SEQGEN
SHAREDMIDDLE
SRECORDS
STGMGT
TSPRT
UOCONST
UODATA
UONLS
UOOSI
UTILITY
VERIFY
WMDATA

(opnstlag)	<nine, msg3format, 01778>	EXT CONSTANT =1030001000008	
(outline)	<nine, msg3format, 01766>	EXT CONSTANT =2	3A9B
(pnam)	<nine, msg3format, 01746>	EXT CONSTANT =1	3A6A1
(pnend)	<nine, msg3format, 01756>	EXT CONSTANT =9	3A6B4
(pnlen)	<nine, msg3format, 01755>	EXT CONSTANT =8	3A6B3
(pnmax)	<nine, msg3format, 01754>	EXT CONSTANT =7	3A6B2
(prcelt)	<nine, msg3format, 010>	CONSTANT =2	3C2B
(psisig)	<nine, msg3format, 01760>	EXT CONSTANT =20B	3A7B
(qwait)	<nine, msg3format, 01757>	EXT CONSTANT =6	3A6B5
(rawpcp)	<nine, msg3format, 01729>	EXT CONSTANT =5	3A3C
(rcvchannel)	<nine, msg3format, 01771>	EXT CONSTANT =4	3A10A2
(rcvdone)	<nine, msg3format, 01721>	CONSTANT =3	4C1C
(rcvgeneric)	<nine, msg3format, 01734>	EXT CONSTANT =4	3A4D
(rcvspecific)	<nine, msg3format, 01733>	EXT CONSTANT =3	3A4C
(rcvstart)	<nine, msg3format, 01722>	CONSTANT =2	4C1B
(resisize)	<nine, msg3format, 01174>	CONSTANT =30	3C4
(sbrh)	<nine, msg3format, 0842>	FIELD - 8	4A5B1
(sendalarm)	<nine, msg3format, 01735>	EXT CONSTANT =5	3A4E
(sendgeneric)	<nine, msg3format, 01732>	EXT CONSTANT =2	3A4B
(sendspecific)	<nine, msg3format, 01731>	EXT CONSTANT =1	3A4A
(seghndl)	<nine, msg3format, 01742>	EXT CONSTANT =1	3A5B
(sig)	<nine, msg3format, 01747>	EXT CONSTANT =2	3A6A2
(sigdata)	<nine, msg3format, 01795>	FIELD - 18	3B2A
(signalr)	<nine, msg3format, 01794>	RECORD	3B2
(sixtnbitr)	<nine, msg3format, 0841>	RECORD	4A5B
(sndmsg)	<nine, msg3format, 01425>	PROCEDURE	4C4
(sndmsg)	<nine, msg3format, 01506>	PROCEDURE	4B2
(sndstart)	<nine, msg3format, 01718>	CONSTANT =1	4C1A
(sphndl)	<nine, msg3format, 01751>	EXT CONSTANT =6	3A6A6
(sppcp)	<nine, msg3format, 01727>	EXT CONSTANT =1	3A3A
(srcnam)	<nine, msg3format, 01753>	EXT CONSTANT =1	3A6B1
(stcbit)	<nine, msg3format, 01774>	EXT CONSTANT =2B10	3A10A5
(stcchannel)	<nine, msg3format, 01770>	EXT CONSTANT =4	3A10A1
(stcmtopr)	<nine, msg3format, 01772>	EXT CONSTANT =4B6	3A10A3
(stopme)	<nine, msg3format, 01739>	EXT CONSTANT =101	3A4I
(tidelt)	<nine, msg3format, 011>	CONSTANT =3	3C2C
(ttyal)	<nine, msg3format, 0994>	PROCEDURE	4A5H
(ttyas)	<nine, msg3format, 0999>	PROCEDURE	4A5I
(ttych)	<nine, msg3format, 01004>	PROCEDURE	4A5J
(typelt)	<nine, msg3format, 09>	CONSTANT =1	3C2A
(typemessg)	<nine, msg3format, 0939>	PROCEDURE	4A5F
(typemsg)	<nine, msg3format, 01620>	PROCEDURE	4C5
(typemsg)	<nine, msg3format, 0844>	PROCEDURE	4B4
(typen)	<nine, msg3format, 01826>	PROCEDURE	4C7
(typepcp)	<nine, msg3format, 0960>	PROCEDURE	4A5G
(typeprsnam)	<nine, msg3format, 0920>	PROCEDURE	4A5E
(typesig)	<nine, msg3format, 01814>	PROCEDURE	4C6
(unblock)	<nine, msg3format, 01759>	EXT CONSTANT =40B	3A7A
(utoptype)	<nine, msg3format, 01118>	PROCEDURE	4A6C

```
< NINE, MSG3FORMAT.NLS;12, >, 15-May-78 13:35 SKO ;;;;
FILE %needed for LIBRARY subsystem%
% Branches to include in interface source modules using MSG3
transaction  format for messages %
% Declarations %                                                    3
   %constants%
      (atbmty) EXTERNAL CONSTANT = 440800B6; %points before first 8 bit
      byte%                                                         3A1
      (msgwait) EXTERNAL CONSTANT = 24*60*60000; %milleseconds for
      timeout%                                                      3A2
      % "howpcp" values %
         (sppcp) EXTERNAL CONSTANT = 1;  %AC0 at startup if shared
         page%                                                      3A3A
         (dpspcp) EXTERNAL CONSTANT = 2;  %AC0 at startup if DPS
         protocol%                                                  3A3B
         (rawpcp) EXTERNAL CONSTANT = 5;  %AC0 at startup if raw
         network%                                                   3A3C
         (nswdpcp) EXTERNAL CONSTANT = 10;  %NSW direct connecti n% 3A3D
      %msg primitive constants%
         (sendspecific) EXTERNAL CONSTANT = 1;                      3A4A
         (sendgeneric) EXTERNAL CONSTANT = 2;                       3A4B
         (rcvspecific) EXTERNAL CONSTANT = 3;                       3A4C
         (rcvgeneric) EXTERNAL CONSTANT = 4;                        3A4D
         (sendalarm) EXTERNAL CONSTANT = 5;                         3A4E
         (enablealarm) EXTERNAL CONSTANT = 6;                       3A4F
         (openconn) EXTERNAL CONSTANT = 7;                          3A4G
         (closeconn) EXTERNAL CONSTANT = 8;                         3A4H
         (stopme) EXTERNAL CONSTANT = 101;                          3A4I
      %msg handling constants %
         (nohndl) EXTERNAL CONSTANT = 0;                            3A5A
         (seqhndl) EXTERNAL CONSTANT =1;                            3A5B
         (markhndl) EXTERNAL CONSTANT =2;                           3A5C
      %msg param constarts%
         %sendspecfic and sendgeneric%
            (pnam) EXTERNAL CONSTANT = 1;                           3A6A1
            (sig) EXTERNAL CONSTANT = 2;                            3A6A2
            (disp) EXTERNAL CONSTANT =3;                            3A6A3
            (dt) EXTERNAL CONSTANT =4;                              3A6A4
            (msgbytes) EXTERNAL CONSTANT =5;                        3A6A5
            (sphndl) EXTERNAL CONSTANT =6;                          3A6A6
         %receivespecific and generic%
            (srcnam) EXTERNAL CONSTANT = 1;                         3A6B1
            (pnmax) EXTERNAL CONSTANT = 7; %in bytes%               3A6B2
            (pnlen) EXTERNAL CONSTANT = 8;                          3A6B3
            (pnend) EXTERNAL CONSTANT = 9;                          3A6B4
            (qwait) EXTERNAL CONSTANT = 6;                          3A6B5
      %signal types%
         (unblock) EXTERNAL CONSTANT = 40B;                         3A7A
         (psisig) EXTERNAL CONSTANT = 20B;                          3A7B
      % IPC call and return constants%
         (call) EXTERNAL CONSTANT = 1;                              3A8A
         (ack) EXTERNAL CONSTANT = 2;                               3A8B
      % Callmode for call on external procedure %
         (inline) EXTERNAL CONSTANT = 1;  %inline with acknowledgement%
                                                                    3A9A
         (outline) EXTERNAL CONSTANT = 2;  %out-of-line with
```

```
    acknowledgement%                                                    3A9B
    (noack) EXTERNAL CONSTANT = 3;  %out-of-line without
    acknowledgement%                                                    3A9C
  %direct connection constants%
    %connection psi channel data%
      (stcchannel) EXTERNAL CONSTANT = 4;  % connection state
      change channel%                                                   3A10A1
      (rcvchannel) EXTERNAL CONSTANT = 4;  %BE recevie channel%
                                                                        3A10A2
      (stcmtopr) EXTERNAL CONSTANT = 4B6;  %state change channel
      in bits 12-17 - AC3 for mtopr%                                    3A10A3
      (intmtopr) EXTERNAL CONSTANT = 5B10;  %interrupt channel in
       bits 0-5 - AC3 for mtopr%                                        3A10A4
      (stcbit) EXTERNAL CONSTANT = 2B10;  %state change channel
      is bit 4 - AC2 for IIC%                                           3A10A5
      (msgpriority) EXTERNAL CONSTANT = 2;  %interrupt priority%
                                                                        3A10A6
    %OPENF flags for network connections - 8-bit bytes, unblocked
    mode%
      (opnrflag) EXTERNAL CONSTANT = 1030002000008;  %receive%
                                                                        3A10B1
      (opnsflag) EXTERNAL CONSTANT = 1030001000008;  %send% 3A10B2
    % GTFJN flag for network connections%
      (gjfnflag) EXTERNAL CONSTANT = 1B6;  %get jfn flag%   3A10C1
    % job log in paramters %
      (logdir) EXTERNAL CONSTANT = 1477B;  %OCKEN directory
      number%                                                           3A10D1
      (logpas) EXTERNAL STRING = "SKO";  %OCKEN  directory
      password%                                                         3A10D2
      (logacc) EXTERNAL STRING = "SRI";  %OCKEN directory
      account%                                                          3A10D3
    %string of local socket for listen for receive - job relative%
      (lrstr) EXTERNAL STRING = "NET:0.;T";                             3A10E1
    %string of local socket for listen for send - job relative%
      (lsstr) EXTERNAL STRING = "NET:1.;T";                             3A10F1
  %error log constants%
    (gjfneflag) EXTERNAL CONSTANT = 1B6;  %gtjfn flag - use high
    version or create new%                                              3A11A
    (opneflag) EXTERNAL CONSTANT = 0700000200008;  %openf flag -
    append%                                                             3A11B
    (errfile) EXTERNAL STRING = "<relnine>errlog.txt";                  3A11C
  % bitstr conversion constants %
    DECLARE % CAUTION: Don't change the order %
      lft _ (
        % lft32 - 0 % 0440B8 ,
        % lft28 - 1 % 1034B8 ,
        % lft24 - 2 % 1430B8 ,
        % lft20 - 3 % 2024B8 ,
        % lft16 - 4 % 2420B8 ,
        % lft12 - 5 % 3014B8 ,
        % lft8  - 6 % 3410B8 ,
        % lft4  - 7 % 4004B8),
      mid _ (
        % mid32 - 0 % 0440B8 ,
        % mid28 - 1 % 0434B8 ,
        % mid24 - 2 % 0430B8 ,
```

```
            % mid20 - 3 % 0424B8 ,
            % mid16 - 4 % 0420B8 ,
            % mid12 - 5 % 0414B8 ,
            % mid8  - 6 % 0410B8 ,
            % mid4  - 7 % 0404B8),
         rit _ (
            % rit4  - 0 % 0004B8 ,
            % rit8  - 1 % 0010B8 ,
            % rit12 - 2 % 0014B8 ,
            % rit16 - 3 % 0020B8 ,
            % rit20 - 4 % 0024B8 ,
            % rit24 - 5 % 0030B8 ,
            % rit28 - 6 % 0034B8 ,
            % rit32 - 7 % 0040B8);
%records%
   (dbrec) RECORD                                                    3B1
      dbprim[3], % TRUE => output primitive info %
      dbparam[3], % TRUE => output parameter info %
      dbprs[3], % TRUE => output process name info %
      dbmsg[3]; % TRUE => output message info %

   (signalr) RECORD %tenex signal record%                            3B2
      sigdata[18], sigtype[6], sigevent[12];
% Declarations %
   % External labels %
      EXTERNAL logjob;
      EXTERNAL errlog;
      EXTERNAL invmsg;
   % Indices into message list (L10 list) - shared page protocol %
      (typelt) CONSTANT = 1;  %message type - call/acknowledgement%
                                                                     3C2A
      (prcelt) CONSTANT = 2;  %procedure name/error code%            3C2B
      (tidelt) CONSTANT = 3;  %transaction identifier or 0%          3C2C
      (argelt) CONSTANT = 4;  %argument/result list%                 3C2D
   % NSW constants %
      (nswrer) CONSTANT = 4;  %NSW recoverable error%                3C3A
   (reslsize) CONSTANT = 30;  %used to allocated FE results list% 3C4
   (bpt32r) CONSTANT = 401000000000B;  %byte ptr-32 right most bits%
                                                                     3C5
%source-code%                                                          4
   %common to shared page, raw network, MSG3 direct connection and MSG3
   messages%                                                          4A
      % initialization, main control loop , and external call routines
      %
         (execmiddle)   % GB: ;  Main control loop %
         PROCEDURE;                                                  4A1A
            % Procedure description
               FUNCTION
                  This is the version of the main control loop using a
                  MSG3 or shared page protocol as the communication to
                  the frontend.
               ARGUMENTS
                  none
               RESULTS
                  proc-value
               NON-STANDARD CONTROL
```

```
            none
         GLOBALS
            none
         %
      % Declarations %
         LOCAL tid, pcpport, errorcode;
      % Loop forever getting and processing messages (calls on BE
      procedures) %
         LOOP
            handlemsg();
      % Return %
         RETURN;
      END.

   (extcall)   % GB: ;  call remote process MSG3/Shared Page %
   PROCEDURE (dest, pname REF, callmode, argl REF % =>
   calloutcome, resl %);                                          4A1P
      % Procedure description
         FUNCTION
            Set up message to call procedure in remote process.
            Send message.  If acknowledgement is requested,
            process return message .  DOES NOT PROCESS
            OUT-OF-LINE CALLS WITH ACKNOWLEDGEMENT
            Check flag 'dssflag' (disable show status flag).  If
            TRUE and call is to SHOWSTATUS in FE, RETURN without
            calling FE.
            Shared page
               This routine assumes that 'argl' is a list in the
               list zone.  It does not copy the list.
               If there is no tool result conversion routine,
               'resl' is set (in decmsg) to the address of the
               list of results set up by the FE.  If there is a
               tool result conversion routine, 'resl' is set to
               point to the BE global list for results ('exresl')
            MSG3 or raw network
               'resl' is set to point to the BE global list for
               results ('exresl')
         ARGUMENTS
            dest:  destination process ???
            pname:  address of string containing procedure name
            callmode:  mode of message
               values:
                  noack - no acknowledgement
                  inline - in line with acknowledgement
                  outline - out of line with acknowledgement
            argl :  address of argument list
         RESULTS
            calloutcome:  TRUE if call successful, FALSE if
            failure
               Note: the calling routine does not get the actual
               error code from the FE (errorcode) but only a t/f
               flag
            resl :  address of result list
         NON-STANDARD CONTROL
            none
         GLOBALS
```

```
        exresl:  nulls and sets
      %
    % Declarations %
      LOCAL cnvsuc, calloutcome _ TRUE, i, errorcode, tid,
      acktid, pcpport, msglen, reslptr REF, resl REF;
      LOCAL STRING cerrstr[50], tracestr[100];
      LOCAL LIST rawresults[reslsize];
      REF exresl;
    % Check disable show status flag%
      IF dssflag AND (*pname* = "SHOWSTATUS" OR *pname* =
      "showstatus")
        THEN  RETURN;
    %Invoke catch phrase%
      INVOKE(catext);
    % Set up list for results %
      #exresl# _ ;  %null out from last call to FE%
      &resl _ &exresl;
    %trace args%
      IF mdebug .A atr THEN
        BEGIN
        *tracestr* _ "Arguments for remote procedure ",
        *pname*, ":";
        traceparams($tracestr, &argl);
        END;
    % Set up and send message %
      CASE callmode OF
        = noack:  %no acknowledgement%
          tid _ 0;
        = inline:  %in-line with acknowledgment%
          BEGIN
          %ASSIGNMENT OF TID MUST BE DIFFERENT WHEN
          OUT-OF-LINE CALLS WITH ACKNOWLEDGEMENT IS
          IMPLEMENTED%
          tid _ callct _ callct + 1;
          END;
        = outline:  %out-of-line with acknowledgement%
          badmsg($"out-of-line calls with acknowedgement not
          implemented - extcall");
        ENDCASE badmsg($"invalid callmode - extcall");
      %Enclode message using an empty list in place of a zero
      for argument list %
        msglen _ encmsg(call, &pname, tid,
          IF &argl THEN &argl ELSE &resl  %empty list%);
      sndmsg( msglen );  %  Send message %
    % Process results if acknowedgement requested %
      IF tid AND callmode = inline THEN
        BEGIN
        %THIS CODE ASSUMES CALLS ARE ACKNOWLEDGED IN THE
        ORDER THAT THEY ARE SENT.  NEED DIFFERENT PROCESSING
        IF THIS IS NOT THE CASE%
        UNTIL handlemsg (acktid) = ack DO;
        BUMP DOWN callct;  %decrement outstanding call count%
        errorcode _
          decmsg(0, IF toolrcnv THEN $rawresults ELSE &resl:
          &reslptr);
        calloutcome _ IF errorcode THEN FALSE ELSE TRUE;
```

```
            %go to tool's result conversion routine%
               IF toolrcnv THEN
                  BEGIN
                  FOR i _ 1 UP UNTIL > reslptr.L DO
                     BEGIN
                     [toolrcnv](&reslptr, &resl, i, $cerrstr :
                     [cnvsuc] );
                     IF NOT cnvsuc THEN
                     ABORT(ermunknown,$cerrstr);
                     END;
                  END
               ELSE &resl _ &reslptr;  %list is completely
               processed%
            %trace results%
               IF mdebug .A rfr THEN
                  BEGIN
                  *tracestr* _ "Converted results from remote
                  procedure ", *pname*, ":";
                  traceparams($tracestr, &resl);
                  END;
            END;
      % Return %
         #rawresults# _ ;
         DROP (catext);
         RETURN(calloutcome, &resl);
      % Catchphrases %
         (catext)  CATCHPHRASE();                                  4A1B10A
            CASE SIGNALTYPE OF
               = notetype:
                  CASE SIGNAL OF
                     = return, = unwind :
                        BEGIN  %free space for allocated lits%
                        DISABLE (catext);
                        #rawresults# _ ;
                        END;
                  ENDCASE CONTINUE;
            ENDCASE CONTINUE;
      END.

   (intext)   % GB ;  set globals for calling external procedures
   %
   PROCEDURE ;                                                     4A1C
      % Procedure description
         FUNCTION
            set globals that will be used as parameters to
            extcall.
         ARGUMENTS
            none
         RESULTS
            proc-value
         NON-STANDARD CONTROL
            none
         GLOBALS
            sets:  feident, fetoolpkg, fedpypkg
         %
      % Declarations %
```

```
        % set globals for calls on extcall %
           % parameters to extcall for calls on FE - callmode is
           synchronous with acknowedgement %
              fetoolpkg _ inline;  % used as callmode parameter %
              fedpypkg _ inline;  % used as callmode parameter %
              feident _ femailbox[1];
           % VARIABLES FOR CALLS ON OTHER TOOLS %
        % Return %
           RETURN;
        END.

  % Sending and Receiving message routines %
     (handlemsg)    % LB: ;  get MSG3/Shared page message %
     PROCEDURE % => type, tid %;                                    4A2A
        % Procedure description
           FUNCTION
              Get message and determine its type.  Could be MSG3 or
              L10 list message.  For MSG3  begin processing using
              RLIST.  Go to docall if message is call on BE
              procedure.
              NOTE: THIS ROUTINE ASSUMES THAT THERE ARE NO
              OUT-OF-LINE CALLS WITH ACKNLEDGEMENT AND THAT CALLS
              ARE ACKNOWLEDGED IN THE ORDER THAT THEY OCCURED
           ARGUMENTS
              none
           RESULTS
              type:  type of message
              tid:  transaction identifier
           NON-STANDARD CONTROL
              none
           GLOBALS
              rcvlst, typelt, prcelt, argelt:  references
           %
        % Declarations %
           LOCAL type, tid, pcpport;
        % Get message %
           getmsg();
        %Process message for type and tid%
           CASE howpcp OF
              = sppcp:  %L10 list%
                 BEGIN
                 type _ ELEM #rcvlst#[typelt];  %get type%
                 tid _ ELEM #rcvlst#[tidelt];  %get tid%
                 % If call, go to docall %
                    CASE type OF
                       = call: %call on routine in BE%
                          docall(tid);
                       = ack: %acknowledgement of call on FE%
                          BEGIN
                          % Check that this is acknowledgment for
                          current call %
                          IF tid NOT= callct THEN
                             badmsg($"Bad tid in proc return");
                          END;
                       ENDCASE
                          badmsg($"Bad message type");
```

```
                END;
            ENDCASE  %MSG3 message%
                BEGIN
                OPENPORT rlist($inrcvbuf, $dsphlk : [pcpport]);
                PCALL [pcpport] (pcplist, 4);
                PCALL [pcpport] (pcpindex, 0 :type); %get type%
                PCALL [pcpport] (pcpindex, 0 : tid); %get tid for
                call%
                % If call, go to docall %
                   CASE type OF
                      = call: %call on routine in BE%
                         docall(tid);
                      = ack: %acknowledgement of call on FE%
                         BEGIN
                         %This code is commented out.  It is
                         needed to implement out-of-line calls
                         with acknowledgment
                            IF tid IN [1, maxch] THEN %%free
                            outstanding call rec%%
                               [$outstanding + (tid -1)
                               *ocallrec.SIZE].outgram _ 0
                            ELSE
                               badmsg($"Bad tid in proc return");
                         %
                         % Check that this is acknowledgment for
                         current call %
                         IF tid NOT= callct THEN
                            badmsg($"Bad tid in proc return");
                         END;
                      ENDCASE
                         badmsg($"Bad message type");
                END;
        % Return %
           RETURN(type, tid);
        END.

% Misc. routines %
   (ipcinit)   % LB: ;  MSG3 dependent initialization %
   PROCEDURE;                                                    4A3A
      % Procedure description
        FUNCTION
             MSG3 dependent initialization
        ARGUMENTS
           none
        RESULTS
           proc-value
        NON-STANDARD CONTROL
           none
        GLOBALS
           none
        %
      % Declarations %
      % Return %
        RETURN;
      END.
```

```
(docall)   % LP: ;  call from remote process on backend
routine %
PROCEDURE ( tid );                                              4A3B
   % Procedure description
      FUNCTION
         Here when a MSG3 or shared page message is a call.
         This procedure decodes the message , calls DISPATCH
         to call the BE routine, and sends a message to the
         calling process if acknowledgment was requested.  For
         shared page protocol, it copies the x-routine results
         to a list in the list zone before sending the results
         to the FE.
      ARGUMENTS
         tid:  transaction id
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         exresl:  sets and nulls
         dspcmd:  nulls on ABORT
      %
   % Declarations %
      LOCAL outcome,  %meta result from x-routine - ignored%
      argptr REF,
      msglen,
      errorcode _ 0,  %set at callfail - sent to FE%
      errstr REF;  %set at callfail - first result to FE%
      LOCAL STRING procname[50];
      LOCAL LIST results[reslsize],args[reslsize];
      REF dspcmd, exresl;
   % Invoke catchphrases %
      INVOKE (callfail, docrtn);
      INVOKE (catdocall);
   %decode message%
      decmsg($procname, $args: &argptr);
   % Call procedure %
      dispatch(0, $procname, &argptr, $results: [outcome]);
   % If acknowledgement  requested, encode and send results %
      (docrtn):                                                 4A3B6A
      IF tid THEN
         BEGIN
            CASE howpcp OF
               = sppcp:  %copy results into shared pages%
                  BEGIN
                  #exresl# _ COPY #results#;
                  &argptr _ &exresl;  %point to list%
                  END;
                ENDCASE  %don't need to copy list%
                  &argptr _ $results ;  %point to list%
         msglen _ encmsg(ack, errorcode, tid, &argptr);
         sndmsg(msglen);
         END
      ELSE IF errorcode THEN
         NULL;  %SHOULD DO SOMETHING LIKE SEND AN ALARM IF
         FAILURE AND NO ACKNOWLEDGEMENT REQUESTED%
```

```
      % Drop catchphrase and free list storage %
        DROP (ALL);
        #results# _ ;
        #args# _ ;
      % Return %
        RETURN;
      % Catchphrases %
        (callfail)  CATCHPHRASE(:&errstr);                          4A3B9A
          CASE SIGNALTYPE OF
            = aborttype:
              BEGIN
              DISABLE (callfail);
              #dspcmd# _ ;  %in case zone out of space%
              IF SIGNAL < 140018 OR SIGNAL > 16777B THEN
                BEGIN
                errorcode _ CASE SIGNAL OF
                  = nohelp : cnohelp;
                  = return : creturn;
                  = unwind : cunwind;
                  = gothelp : cgothelp;
                  = stringoverflow : zstringoverflow;
                  = changestring : cchangestring;
                  = saroverflow : csaroverflow;
                  = stkoverflow : cstkoverflow;
                  = stkunderflow : cstkunderflow;
                  = listspace : clistspace;
                  = programbug : cprogrambug;
                  = uncaughtabort : cuncaughtabort;
                  ENDCASE unknownerr;
                END
              ELSE errorcode  _ SIGNAL;
              IF mdebug THEN mtypeas(&errstr);
              #results# _ *errstr*;  %first result is error
              message%
              TERMINATE;
              END;
          ENDCASE CONTINUE;
        (catdocall)  CATCHPHRASE;                                   4A3B9B
          CASE SIGNALTYPE OF
            = notetype:
              CASE SIGNAL OF
                = return, = unwind :
                  BEGIN
                  DISABLE (catdocall);
                  #results# _ ;
                  #args# _ ;
                  END;
              ENDCASE CONTINUE;
          ENDCASE CONTINUE;
      END.

    (msgerr) %print msg error messg - invoke msg failed or error
    disp%
    PROCEDURE (num);                                                4A3C
      LOCAL STRING msgestr[30];
      *msgestr* _ "MSG error # ", STRING(num);
```

```
      err($msgestr);
      END.

   (badmsg)  %error in encoding or decoding message%
   PROCEDURE(errstr  %error message%);                                  4A3D
      ABORT(ermmsg, errstr);
      END.

   (horrible)   % CL: ;  L10 Runtime recovery procedure %
   PROCEDURE (code, why REF, addr);                                     4A3E
      % Procedure description
        FUNCTION
           Print error message to terminal.  Log error message
           in error log file.  Loop at !haltf.
        ARGUMENTS
           code:  error code
           why:  addresss of error message
           addr: address of location where error occured
        RESULTS
           never returns
        NON-STANDARD CONTROL
           none
        GLOBALS
           none
        %
      % Declarations %
        LOCAL STRING errmsg[200];
      % Write message to terminal %
        IF why.L > 100 THEN why.L _ 100;  %prevent overflow%
        *errmsg* _ "Error code " , STRING (code,8), " address ",
        STRING (addr.RH, 8), " ", *why*;
        !sout(101B, chbmty + $errmsg, -errmsg.L);
        erraddr _ $errmsg;
      % Log message into error log file %
        (errlog):  % here directly if connections never opened%
                                                                        4A3E4A
           % write error message to error log file %
           IF errjfn _ sgtjfn(gjfneflag, $errfile, $lit) THEN
              BEGIN
              IF SKIP !openf(errjfn, opneflag) THEN
                 BEGIN
                 !bout(errjfn, CR);  %start new line%
                 !bout(errjfn, LF);  %start new line%
                 !gjinf();  %get job number%
                 !nout(errjfn, R3, 000005000012B);  %output job
                 no.%
                 !odtim(errjfn, -1, 0);  %output date, time%
                 IF erraddr THEN !sout(errjfn, chbmty +
                 erraddr);
                 !bout(errjfn, CR);  %start new line%
                 !bout(errjfn, LF);  %start new line%
                 IF NOT SKIP !closf(errjfn) THEN NULL;
                 END;
              END;
      % Write message to terimnal and halt %
        LOOP !haltf();
```

```
        !sout(101B, chbmty + $" - HALTF at errlog in NLS BE ", -29
        %length of message%);
        END.

    % Encoding and decoding routines %
      (decmsg)   % LP: ;  decode message - MSG3/Shared Page/Raw
      Network%
      PROCEDURE ( procname REF, args REF LIST );                    4A4A
        % Procedure description
          FUNCTION
            Here to process message parameter (procedure name or
            outcome indicator) and arguments
          ARGUMENTS
            procname:  address of string in which to store
            procedure name
            args:  address of list in which to store
            arguments/results
          RESULTS
            errorcode: errorcode if this is an error return or 0
            if success
            &args:  address of list of arguments/results
          NON-STANDARD CONTROL
            none
          GLOBALS
            rcvlst, typelt, prcelt, argelt:  references
          %
        % Declarations %
          LOCAL type, pcpport, pstring REF, errorcode, listsize,
          i;
        % Check protocol and decode message %
          CASE howpcp OF
            = sppcp:  %shared page%
              BEGIN  %results are in rcvlst%
                type _ ELEM #rcvlst#[typelt];
                CASE type OF
                  = call:  %call on BE proc.  get proc name%
                    BEGIN
                    &pstring _ ELEM #rcvlst#[prcelt];
                    *procname* _ + SF(*pstring*)
                    SE(*pstring*);
                    END;
                  ENDCASE  %acknowledgement%
                    errorcode _ ELEM #rcvlst#[prcelt];
                &args _ ELEM #rcvlst#[argelt];
              END;
            ENDCASE  %MSG3 or raw newtwork%
              BEGIN
              % Open port and get type %
                OPENPORT rlist($inrcvbuf, $dspblk : [pcpport]);
                PCALL [pcpport] (pcplist, 4);
                PCALL [pcpport] (pcpindex, 0 :type); %get type%
              % Process parameter %
                PCALL [pcpport] (rlistignore , 1); %skip tid %
                CASE type OF
                  = call:  %call on BE procedure.  get
                  procedure name%
```

```
                        BEGIN
                        PCALL [pcpport] (pcpcharstr, 0:
                        &pstring);
                        %convert to upper case and store%
                           *procname* _ + SF(*pstring*)
                           SE(*pstring*);
                        freeblk(&pstring, $dspblk);  %release
                        storage%
                        END;
                     = ack:  %return from FE procedure.  get
                     error list%
                        BEGIN  %there are args/results%
                        PCALL [pcpport] (pcpany: listsize);
                        IF listsize THEN
                           BEGIN  %error - get FE error code%
                           PCALL [pcpport] (rlistignore , 1);
                           %skip error class %
                           PCALL [pcpport] (pcpindex, 0
                           :errorcode);
                           END;
                        errorcode _ 0;  %success%
                        END;
                     ENDCASE badmsg($"Bad message type -
                     decmsg");
               % Process arguments/results - listsize with be 0
               if none %
                  PCALL [pcpport] (pcpany: listsize);
                  FOR i _ 1 UP UNTIL > listsize DO
                     #args#[i] _ USE decstruc(pcpport);  %convert
                     one element%
               END;
   % Return %
      RETURN(errorcode, &args);
   END.

(encmsg)   % LB: ;  encode message -
MSG3/SharedPage/RawNetwork %
PROCEDURE (type, param REF, tid, results REF LIST % => msglen
%);                                                          4A4B
   % Procedure description
      FUNCTION
         Set up a message.
            Shared page,
               sets up list in 'sndlst'.
               assumes 'results' is a list in the list zone
            MSG3 or raw network
               uses WLIST.
      ARGUMENTS
         type:  message type
            values:
               call - call on procedure in remote process
               ack - acknowledgement of call on BE procedure
         param:  error code if acknowledgment; address of
         string containing procedure name if external call
         tid:  transaction identifier / 0
         results:  results from BE procedure or arguments for
```

```
            external call
         RESULTS
            msglen:  length of message (no. of bytes)
         NON-STANDARD CONTROL
            Call badmsg if wlist buffer overflows - badmsg
            ABORT's
         GLOBALS
            none
         %
      % Declarations %
         LOCAL pcpport, lenadr, msglen;
      %make message%
         CASE howpcp OF
            = sppcp:  %shared page%
               BEGIN  %L10 list%
               CASE type OF
                  = call:  %call on Fe%
                     BEGIN
                     #sndlst# _ USE makedesc( uindex, type,
                     FALSE),
                        *param*,  %procedure name%
                        USE makedesc(uindex, tid, FALSE),
                           %transaction identifier or 0%
                        USE makedesc(ulist, &results, FALSE);
                     END;
                  = ack:  %acknowledgement%
                     BEGIN
                     #sndlst# _ USE makedesc( uindex, type,
                     FALSE),
                        USE makedesc(uindex, &param, FALSE),
                        USE makedesc(uindex, tid, FALSE),
                           %transaction identifier or 0%
                        USE makedesc(ulist, &results, FALSE);
                     END;
                  ENDCASE badmsg($"Bad message type - encmsg");
               END;
            ENDCASE  %MSG3 or raw network%
               BEGIN
               OPENPORT wlist($sndbuf, sbsiz : [pcpport]);
               INVOKE (sendovrflow);
               PCALL [pcpport] (pcplist, 4);
               PCALL [pcpport] (pcpindex, type);
               PCALL [pcpport] (pcpindex, tid); %tid%
               CASE type OF
                  = call:  %call on external procedure%
                     PCALL [pcpport] (pcpcharstr, &param);
                     %procedure name%
                  = ack:  %acknowledgement from BE procedure%
                     BEGIN
                     IF &param THEN
                        BEGIN  % set up 3 element list for error
                        %
                        PCALL [pcpport] (pcplist, 3);
                        PCALL [pcpport] (pcpindex, nswrer);
                           %error class - NSW recoverable%
                        PCALL [pcpport] (pcpindex, &param);
```

```
                        %error code%
                     PCALL [pcpport] (pcpcharstr, ELEM
                     #results#[1]);
                        %error message%
                     END
                  ELSE PCALL [pcpport] (pcplist, 0);
                  %success%
                  PCALL [pcpport] (wlistend);  %close list%
                  END;
               ENDCASE badmsg($"Bad message type - encmsg");
            encstruc(&results, pcplist, pcpport); % convert
            and store results / arguments %
            %close message list - get number of bytes in
            message%
               msglen _ PCALL [pcpport] (wlistend);
            DROP (sendovrflow);
            END;
      % Return %
         RETURN(msglen);
      % Catchphrases %
         (sendovrflow) CATCHPHRASE;                          4A4B5A
            CASE SIGNALTYPE OF
               = helptype :
                  IF SIGNAL = wlistoverflow THEN
                     badmsg($"WLIST overflow - encmsg");
            ENDCASE;
      END.

% MSG-3 Debugging procedures %
   %+DEBUG%                                                  4A5A
   (sixtnbitr) RECORD % 8 bit bytes from rightmost 16 bits record
   %                                                         4A5B
      sbrh[8], sblh[8];

   DECLARE STRING mstring[4800];

   (addsig) %add msg-3 signal to string%
   PROCEDURE (pb REF);                                       4A5D
      CASE pb[sig].sigtype OF
         = msgpsi:
            *mstring* _ "msgpsi chnl ", STRING(pb[sig].sigdata,
            8), "B ";
         = unblock:
            *mstring* _ "unblock ";
         ENDCASE err($"Bad signal in MSG-3 -TYPESIG");
      *mstring* _ "sig: ", *mstring*;
      RETURN;
      END.

   (typeprsnam) %type a msg-3 process name%
   PROCEDURE (bp %byteptr to first byte of name%);           4A5E
      LOCAL had _0, hinc _0, pins _0, i, maxbytes;
      LOCAL STRING string[80];
      had.sblh _ ^bp;
      had.sbrh _ ^bp;
      hinc.sblh _ ^bp;
```

```
        hinc.sbrh _ ^bp;
        pins.sblh _ ^bp;
        pins.sbrh _ ^bp;
        maxbytes _ MIN(^bp, 40);
        FOR i _ 0 UP UNTIL >= maxbytes DO
           *string* _ *string*, ^bp;
        IF had THEN %put in host addr and incarn #%
           *string* _ "Process name: [", STRING(had, 8), "B] #",
           STRING(hinc),"   " , *string*, " #", STRING(pins)
        ELSE
           *string* _ "Process name: ", *string*, " #",
           STRING(pins);
        ttyal($string);
        RETURN;
        END.

    (typemessg) % type out a messg-3 message%
    PROCEDURE (messg REF);                                          4A5F
        LOCAL pcpport, rtype, value;
        OPENPORT rlist( &messg, $dspblk : [pcpport]);
        ttyas($"Message : ");
        rtype _ PCALL [pcpport] (pcpany : value);
        IF (rtype = pcplist) AND (value = 5) THEN %well-formed
        message%
           BEGIN
           *mstring* _ *mstring*,"TYPE= "; typepcp( pcpport,
           $mstring);
           *mstring* _ *mstring*,"LENGTH= "; typepcp( pcpport,
           $mstring);
           *mstring* _ *mstring*,"TID= "; typepcp( pcpport,
           $mstring);
           *mstring* _ *mstring*,"PARAM= "; typepcp( pcpport,
           $mstring);
           *mstring* _ *mstring*,"ARGS= "; typepcp(pcpport,
           $mstring); ttyal($mstring);
           END
        ELSE
           BEGIN
           *mstring* _ "BAD FORMAT: type: ", STRING(rtype), "
           value: ", STRING(value);
           ttyal($mstring);
           END;
        *mstring* _ NULL;
        RETURN;
        END.

    (typepcp) % put a pcp element into a string%
    PROCEDURE (srceport, string REF);                               4A5G
        LOCAL i, rtype, value;
        CASE rtype _ PCALL [srceport] (pcpany : value) OF
           = pcplist:
              BEGIN
              *string* _ *string*, "L:", STRING(value), "( ";
              FOR i _ 0 UP UNTIL >= value DO
                 typepcp( srceport, &string);
              *string* _ *string*, ") ";
```

```
            END;
         = pcpcharstr :
            BEGIN
            *string* _ *string*, "S:", *[value]*, " ;
            freeblk(value, $dspblk);
            END;
         = pcpindex : *string* _ *string*, "I:",STRING(value), "
         ;
         = pcpboolean : *string* _ *string*, "B:",STRING(value),
         " ;
         = pcpempty : *string* _ *string*, "E: ";
         = pcpinteger :
            BEGIN
            *string* _ *string*, "N:",STRING([value]), " ;
            freeblk(value, $dspblk);
            END;
         = pcpbitstr :
            BEGIN
            *string* _ *string*, "BITS:", STRING([value]), "( ";
            FOR i _ 1 UP UNTIL > ((value]+(WORD-1))/WORD) DO
               *string* _ *string*, " , STRING( [value + i], 8);
            *string* _ *string*, ") ";
            freeblk(value, $dspblk);
            END;
      ENDCASE;
   RETURN;
   END.

(ttyal) %type a line in tty window%
PROCEDURE (string REF);                                          4A5H
   ttyas(&string);
   ttych(EOL);
   RETURN;
   END.

(ttyas) % Type a string%
PROCEDURE (astrng REF);                                          4A5I
   IF NOT astrng.L THEN RETURN;
   !sout(101B, chbmty + &astrng, -astrng.L);
   RETURN;
   END.

(ttych) % Type a character%
PROCEDURE (char);                                                4A5J
   !bout(101B, char);
   RETURN;
   END.

%+DEBUG%                                                         4A5K
% data conversion routines - PCPB8 %
(encstruc)   % LB: ;  encode data structure in PCPB-8 format %
PROCEDURE (nlsds REF, type, pcpport);                            4A6A
   % Procedure description
      FUNCTION
         convert one L10 entitity into the corresponding
         PCPB-8 data structure using WLIST
```

```
      ARGUMENTS
         nlsds: value(or address dpending on type) of a L10
         data structure to encode
         type:  pcp type
         pcpport:  WLIST port
      RESULTS
         none
      NON-STANDARD CONTROL
         ABORT if bad PCP type - error code ermptype
      GLOBALS
         none
      %
   % Declarations %
      LOCAL ptr, i;
   %encode structure type%
      CASE type OF
         = pcpempty, = pcpboolean, = pcpindex, = pcpcharstr:
            PCALL [pcpport] (type, &nlsds);
         % must convert bit string to 32 bits/word
         = pcpbitst:
         %
         = pcpinteger:
            BEGIN
            ptr _ &nlsds;  %WLIST needs address of integer%
            PCALL [pcpport] (type, $ptr);
            END;
         %  there is not currently a PCPB-8 block type
         = pcpblock:
         %
         = pcplist:
            BEGIN
            PCALL [pcpport] (type, nlsds.L);  %set up list%
            %convert elements - OK for 0 length list - FOR
            clause will never be executed%
               FOR i _ 1 UP 1 UNTIL > nlsds.L DO
                  encstruc(ELEM #nlsds#[i], utoptype( DESCR
                  #nlsds#[i] ), pcpport);
            PCALL [pcpport] ( wlistend );  %close list%
            END;
         ENDCASE ABORT(ermptype, $"Illegal PCP data type",
         type);
   % Return %
      RETURN;
   END.

(decstruc)   % LB: ;  decode data structure from PCPB-8 to L10
%
PROCEDURE (pcpport % => led, type %);                          4A6B
   % 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 _ aloblk(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);
            %convert each element - OK for 0 length list -
            never executes FOR clause%
               FOR i _ 1 UP UNTIL > &newds DO
                  #ladr#[i] _ USE decstruc(pcpport :type);
            END;
         ENDCASE ABORT(ermptype, $"Illegal PCP data type",
         type);
   % Return %
      RETURN(&led, type);
   END.

(utoptype)   % LB: ;  get PCPB-8 type for L10 data type %
PROCEDURE (lstdesc % => ptype %);                          4A6C
   % Procedure description
      FUNCTION
         get corresponding PCPB-8 data type for a L10 list
         element
      ARGUMENTS
         lstdesc:  element descriptor
      RESULTS
         ptype:  PCPB-8 data type
      NON-STANDARD CONTROL
         ABORT if invalid type
      GLOBALS
         none
      %
   % Declarations %
      LOCAL type, ptype;
   % get PCPB-8 type %
      CASE type _  lstdesc.ledubv OF
         = unull: ptype _ pcpempty;
         = uindex: ptype _ pcpindex;
         = uinteg: ptype _ pcpinteger;
         = ustrin: ptype _ pcpcharstrr;
         = uboole: ptype _ pcpboolean;
         = ubitst: ptype _ pcpbitst;
         = ulist:  ptype _ pcplist;
         ENDCASE ABORT(ermptype, $"Illegal PCP data type",
         type);
   % Return %
      RETURN(ptype);
   END.

(cnv36to32)    % convert bitstr 36 => 32 %
PROCEDURE( oldbitstr REF, newbitstr REF );                 4A6D
   % Procedure description
      FUNCTION
         convert a bitstr given in 36 bit/wrd into a bitstr
         containing only 32 bit/wrd.
      ARGUMENTS
         oldbitstr -- adress of bitstr having 36 bits/wrd
         newbitstr -- adress of bitstr having 32 bits/wrd
         NOTE
```

```
            newbitstr may require more words than oldbitstr.
            Make sure you have enough room as this procedure
            won't check.
            Overwriting is not permitted.
         RESULTS
            none
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      LOCAL lastadrs, oldwrds, from1, to1, from2, to2, i, ofst1,
      ofst2;
      % compute number of words %
         oldwrds _ (oldbitstr+35)/36;
         lastadrs _ &newbitstr + (oldbitstr+31)/32;
      % copy the length %
         newbitstr _ oldbitstr;
      % copy the rest of the bitstr %
         FOR i _ 0 UP UNTIL >= oldwrds DO
            BEGIN
               DIV i/8, ofst1, ofst2;
               from1 _ lft[ofst2]   + ofst1*8 + ofst2 + 1 +
               &oldbitstr;
               to1   _ mid[ofst2]   + ofst1*9 + ofst2 + 1 +
               &newbitstr;
               from2 _ rit[ofst2]   + ofst1*8 + ofst2 + 1 +
               &oldbitstr;
               to2   _ lft[7-ofst2] + ofst1*9 + ofst2 + 2 +
               &newbitstr;
               .to1 _ .from1;
               IF to2.RH <= lastadrs THEN .to2 _ .from2;
            END;
      RETURN;
      END.

   (cnv32to36)    % convert bitstr 32 => 36 %
   PROCEDURE( oldbitstr REF, newbitstr REF );                    4A6E
      % Procedure description
         FUNCTION
            convert a bitstr given in 32 bit/wrd into a bitstr
            containing only 36 bit/wrd.
         ARGUMENTS
            oldbitstr -- adress of bitstr having 32 bits/wrd
            newbitstr -- adress of bitstr having 36 bits/wrd
            NOTE
               Overwriting is not permitted
         RESULTS
            none
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      LOCAL newwrds, from1, to1, from2, to2, i, ofst1, ofst2;
      % compute number of newwrds %
```

```
            newwrds _ (oldbitstr+35)/36;
         % copy the length %
            newbitstr _ oldbitstr;
         % copy the rest of the bitstr %
            FOR i _ 0 UP UNTIL >= newwrds DO
               BEGIN
                  DIV i/8, ofst1, ofst2;
                  from1 _ mid[ofst2]   + ofst1*9 + ofst2 + 1 +
                  &oldbitstr;
                  to1   _ lft[ofst2]   + ofst1*8 + ofst2 + 1 +
                  &newbitstr;
                  from2 _ lft[7-ofst2] + ofst1*9 + ofst2 + 2 +
                  &oldbitstr;
                  to2   _ rit[ofst2]   + ofst1*8 + ofst2 + 1 +
                  &newbitstr;
                  .to1 _ .from1;
                  .to2 _ .from2;
               END;
         RETURN;
         END.

%msg3 message code%                                                    4B
   (getmsg)   % LB: ;  get a message using MSG3 messages %
   PROCEDURE;                                                          4B1
      % Procedure description
         FUNCTION
            Set up the parameter block inrcvbf and issue a
            ReceiveSpecific to MSG3
         ARGUMENTS
            none
         RESULTS
            proc-value
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      % Declarations %
      % Set up parameter block %
         inrcvpb _ $inrcvbuf/pagelen;
         inrcvpb[srcnam] _ $inrcvname + atbmty;
         inrcvpb[sig].sigtype _ unblock;
         inrcvpb[sig].sigevent _ 0;
         inrcvpb[dt] _ msgwait;
         inrcvpb[pnmax] _ 40;
      % Invoke msg to do the ReceiveSpecific %
         invmsg(rcvspecific, $inrcvpb);
      %+DEBUG%                                                         4B1E
      IF msg3debug THEN
         typemsg( $inrcvpb, rcvspecific);
      %+DEBUG%                                                         4B1G
      % Return %
         RETURN;
      END.

   (sndmsg)   % LB: ;  send a message using MSG3 messages %
```

```
PROCEDURE( msglen);                                                    4B2
   % Procedure description
      FUNCTION
         Set up parameter buffer and issue a SendSpecific to the
         FE.
      ARGUMENTS
         msglen:  length of message in bytes
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
   % Set up send parameter buffer %
      sendpb _ $sndbuf/pagelen; %msgarea page number%
      sendpb[sig].sigtype _ unblock;
      sendpb[dt] _ msgwait;
      sendpb[srcnam] _ $inrcvname + atbmty;  %Fe process name set
      by Receive%
      sendpb[msgbytes] _ msglen;
      sendpb[sphndl] _ nohndl;
   % Invoke msg to do the SendSpecific %
      invmsg(sendspecific, $sendpb);
   % Return %
      RETURN;
   END.

(invmsg)   % LB: ;  invoke MSG3 %
PROCEDURE(type, pb REF);                                               4B3
   % Procedure description
      FUNCTION
         Invoke MSG3
      ARGUMENTS
         type:  type of MSG3 primitive
         pb: address of  param block
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         msgerr if msg invoke fails or the return disposition is
         unsuccessful
      GLOBALS
         none
      %
   % Declarations %
   %+DEBUG%                                                            4B3C
   IF msg3debug THEN
      typemsg( &pb, type);
   %+DEBUG%                                                            4B3E
   % Do MSG3 primitive %
      R0 _ type;
      R1 _ &pb;
      IF NOT SKIP !msg() THEN
         msgerr (R2); %bad msg invoke%
      IF pb[disp] > 0 THEN
```

```
        msgerr(pb[disp]);  %bad msg send%
    % Return %
      RETURN;
    END.

(typemsg) %look at MSG-3 invoke or completion%
PROCEDURE (pb REF, type);                                          484
    LOCAL i;
    INVOKE( mstrovrcp, RETURN);
    *mstring* _ NULL;
    ttych(EOL);
    ttyas($"BE MSG-3 trace: ");
    CASE type OF %determine msg prim type%
      = sendgeneric, = sendspecific:
        BEGIN
        CASE type OF
          =sendgeneric: ttyal($"Send generic");
          =sendspecific: ttyal($"Send specific");
          ENDCASE;
        *mstring* _ "pgnum: ",STRING(pb, 8), "B  nambpt:
        ",STRING(pb[pnam].LH, 8), ",,", STRING(pb[pnam].RH,8),
        "B  ";
        addsig(&pb);
        *mstring* _ *mstring*, " dt: ", STRING(pb[dt]), "
        msgbytes: ", STRING(pb[msgbytes]);
        IF type = sendspecific THEN *mstring* _ *mstring*,"
        sphndl: ",STRING(pb[sphndl]);
        ttyal($mstring); *mstring* _ NULL;
        typeprsnam(pb[pnam]);
        typemessg( pb *pagelen); %typeout message%
        END;
      = rcvspecific:
        BEGIN
        ttyas($"Receive specific");
        IF NOT pb[sig].sigevent THEN %rcv not yet invoked%
          BEGIN
          ttyal($" about to be invoked");
          *mstring* _ "pgnum: ",STRING(pb, 8), "B  nambpt:
          ",STRING(pb[pnam].LH, 8), ",,",
          STRING(pb[pnam].RH,8), "B  ";
          addsig(&pb);
          *mstring* _ *mstring*, " dt: ", STRING(pb[dt]), "
          pnmax: ", STRING(pb[pnmax]);
          ttyal($mstring); *mstring* _ NULL;
          END
        ELSE %completed%
          BEGIN
          ttyal($" completed");
          *mstring* _ "pevent: ",STRING(pb[sig].sigevent), "
          disp: ", STRING(pb[disp]), " msgbytes: ",
          STRING(pb[msgbytes]),"  sphndl: ",STRING(pb[sphndl]);
          ttyal($mstring); *mstring* _ NULL;
          typeprsnam(pb[srcnam]);
          typemessg( pb *pagelen); %typeout message%
          END;
        END;
```

```
         ENDCASE
            BEGIN
            *mstring* _ "Unknown MSG primitive: ", STRING(type);
            ttyal($mstring);
            END;
      DROP(mstrovrcp);
      RETURN;
      (mstrovrcp) CATCHPHRASE (:i);                                    4B4I
         CASE SIGNAL OF
            = stringoverflow : IF i = $mstring THEN
               BEGIN
               IF mstring.L > 0 THEN %type it out and clear it%
                  BEGIN
                  ttyal($mstring);
                  *mstring* _ NULL;
                  RESUME(gothelp, $mstring);
                  END
               ELSE %won't fit%
                  BEGIN
                  ttyal($"Trace msg too long - TYPEMSG");
                  TERMINATE;
                  END;
               END;
            ENDCASE;
      END.

%not-msg3 messages (shared page, raw network or MSG3 direct
connection code)%                                                      4C
   % TYPEMSG status types and connection type %
      (sndstart) CONSTANT = 1;                                         4C1A
      (rcvstart) CONSTANT = 2;                                         4C1B
      (rcvdone) CONSTANT = 3;                                          4C1C
      (bincon) CONSTANT = 4;                                           4C1D
   EXTERNAL rcvcomplete;
   (getmsg)   % LB: ;  get a message - direct, network, shared page
   %
   PROCEDURE;                                                          4C3
      % Procedure description
         FUNCTION
            Determine type of connection
            MSG direct connectin or raw network
               Input a message over the direct (either MSG3 direct
               or raw network) connection.  Do a SIN on the input
               jfn irto the global block inrcvbuf.  Ignore any bytes
               that are not preceeded by 4 bytes of 1's.
            Shared page
               Check global to see if message has been sent.  If so,
               return.  Otherwise wait for it (do a !wait until
               pseudo interrupt occurs).
         ARGUMENTS
            none
         RESULTS
            proc-value
         NON-STANDARD CONTROL
            If there has been an error on the connection, goto
            errlog
```

```
      GLOBALS
         Sets:  rcvchecking, rcvrdy
         Uses:  howpcp, connok
      %
   % Declarations %
      LOCAL length, pcpport, byteptr, headercount;
      LOCAL CONSTANT
         bytsperword = 4, % four 8 bit bytes per word %
         lenbpt = 401000B6; % bytepointer to first header byte %
   CASE howpcp OF
      = sppcp:
         BEGIN
         rcvchecking _ TRUE; % indicate in critical section %
         IF NOT rcvrdy THEN  %wait for psi from FE%
            !wait();  %psi routine DEBRKS to rcvcomplete%
         GOTO rcvcomplete;
         (rcvcomplete): rcvrdy _ FALSE;                          4C3C1E
         rcvchecking _ FALSE; % indicate out of critical section
         %
         END;
      ENDCASE  %Direct MSG connection or raw network%
         BEGIN
         % Check for error %
            IF NOT connok THEN
               BEGIN
               erraddr _ $" Error on connection - getmsg ";
               GOTO errlog;  %bypass signals%
               END;
         %+DEBUG%                                                4C3C2C
         IF debug THEN typemsg( rcvstart, $inrcvbuf, binconn,
         msginputjfn, length);
         %+DEBUG%                                                4C3C2E
         % get the four bytes of precautionary header %
            headercount _ 0;
            LOOP
               BEGIN
               !bin(msginputjfn);
               CASE R2 OF
                  =377B: % a header byte %
                     BEGIN
                     BUMP headercount;
                     IF headercount = 4 THEN EXIT LOOP;
                     END;
                  ENDCASE % garbage on connection - look again
                  for start of header %
                     headercount _ 0;
               END;
         % get the four bytes of length %
            length _ 0;  %zero out -- only 32 bits read%
            !sin( msginputjfn, $length + lenbpt, -4 );
         % read rest of message if it will fit in buffer %
            CASE length OF
               <= irbsize * bytsperword: % message will fit -
               read it %
                  !sin( msginputjfn, $inrcvbuf + atbmty, -
                  length);
```

```
                ENDCASE % message won't fit - error %
                   BEGIN
                   erraddr _ $" Message too long - BE getmsg ";
                   GOTO errlog;  %bypass signals%
                   END;
             %+DEBUG%                                              4C3C2H2
             IF debug THEN typemsg( rcvdone, $inrcvbuf, binconn,
             msginputjfn, length);
             %+DEBUG%                                              4C3C2H4
          END;
    % Return %
       RETURN;
    END.

(sndmsg)   % LB: ;  send message - direct, network, shared page %
PROCEDURE (length);                                                4C4
    % Procedure description
       FUNCTION
          Check protocol type (global howpcp)
          direct MSG or raw network connection
             Output a message over the direct (either MSG3 direct
             or raw network) connection.  Do a SOUT on the output
             jfn from the global block sndbuf
          Shared page
             Do a pseudo interrupt on FE receive channel
       ARGUMENTS
          length:  message length
       RESULTS
          proc-value
       NON-STANDARD CONTROL
          Go to errlog if some error has occured on a connection.
       GLOBALS
          Uses connok
       %
    % Declarations %
       LOCAL shiftnum _ 0, pcpport;
       LOCAL header[2]; % 8 bytes of header %
    % Send message according to protocol type %
       CASE howpcp OF
          = sppcp:  %shared page%
             BEGIN
             shiftnum.RH _ -sndchannel; % neg. for right shift %
             R1 _ fhandle; % FE fork handle %
             R2 _ 4B11; % bit 0 on %
             !LSH R2, @shiftnum; % shift to put sndchnlth bit on %
             !iic(); % interrupt Frontend %
             END;
          ENDCASE  %raw network or direct MSG%
             BEGIN
             % Check connection %
                IF NOT connok THEN
                   BEGIN
                   erraddr _ $" Error on connection - sndmsg ";
                   GOTO errlog;  %write to error log and halt%
                   END;
             %+DEBUG%                                              4C4C1B3
```

```
            IF debug THEN typemsg( sndstart, $sndbuf, binconn,
            msginputjfn, length);
            %+DEBUG%                                          4C4C1B5
            % preceed the message bytes by 8 byte header %
               header[0] _ 777777777777B; % 4 bytes of 1
               indicating the beginning of a new message (a
               precaution in case of garbage on the connection) %
               header[1] _ length * 20B; % next 4 bytes gives
               length of the message - shifted to occupy leftmost
               32 bits%
               !sout( msgoutputjfn, $header + atbmty, -8); % send
               the header %
            % send the message %
               !sout( msgoutputjfn, $sndbuf + atbmty, -length);
            END;
      % Return %
         RETURN;
      END.

   (typemsg) %type out a message%
   PROCEDURE ( status, blk REF, type, jfn, length);                4C5
      LOCAL i;
      ttych(EOL);
      ttyas($"BE trace: ");
      CASE type OF %determine msg prim type%
         = sendgeneric, = sendspecific:
            BEGIN
            IF debug.dbprim THEN
               CASE type OF
                  =sendgeneric: ttyal($"Send generic");
                  =sendspecific: ttyal($"Send specific");
                  ENDCASE;
            IF debug.dbparam THEN
               BEGIN
               typesig(&blk);
               ttyas($" dt: "); typen(blk[dt], 10); ttyas($"  bytes:
               "); typen(blk[bytes], 10);
               IF type = sendspecific THEN
                  BEGIN
                  ttyas($" sphndl: ");
                  typen(blk[sphndl], 10);
                  END;
               ttyas($"  pg: "); typen(blk, 8); ttyas($"  nambpt:
               "); typen(blk[pnam], 8);
               ttych(EOL);
               END;
            IF debug.dbprs THEN typeprsnam(blk[pnam]);
            IF debug.dbmsg THEN typemessg( blk * pagelen); %typeout
            message%
            END;
         = rcvspecific:
            BEGIN
            IF debug.dbprim THEN ttyas($"Receive specific");
            CASE status OF
               = rcvstart: % receive about to be invoked %
                  BEGIN
```

```
                IF debug.dbprim THEN ttyal($" about to be
                invoked");
                IF debug.dbparam THEN
                   BEGIN
                   ttyas($"pg: "); typen(blk, 8); ttyas($"
                   nambpt: "); typen(blk[pnam], 8);
                   ttyas($" dt: "); typen(blk[dt], 10); ttyas($"
                   pnmax: "); typen(blk[pnmax], 10);
                   typesig(&blk);
                   ttych(EOL);
                   END;
                END;
             = rcvdone:  %completed%
                BEGIN
                IF debug.dbprim THEN ttyal($" completed");
                IF debug.dbparam THEN
                   BEGIN
                   ttyas($"pevent: "); typen(blk[sig].sigevent,
                   10); ttyas($" disp: "); typen(blk[disp], 8);
                   ttyas($" bytes: "); typen(blk[bytes], 10);
                   ttyas($" sphndl: "); typen(blk[sphndl], 10);
                   ttych(EOL);
                   END;
                IF debug.dbprs THEN typeprsnam(blk[srcnam]);
                IF debug.dbmsg THEN typemessg( blk * pagelen);
                %typeout message%
                END;
             ENDCASE err($"Bad status -TYPEMSG");
          END;
       = binconn: % sending or rcving over binary connection %
          CASE status OF
             = sndstart: %about to send message over connection%
                BEGIN
                IF debug.dbprim THEN ttyas($"Send binary about to
                be done ");
                IF debug.dbparam THEN
                   BEGIN
                   typen(length, 10); ttyas($" bytes to jfn ");
                   typen( jfn, 8);
                   END;
                ttych(EOL);
                IF debug.dbmsg THEN typemessg( &blk);
                END;
             = rcvstart:
                BEGIN
                IF debug.dbprim THEN ttyas($"Receive binary about
                to be done");
                IF debug.dbparam THEN
                   BEGIN
                   ttyas($" from jfn ");
                   typen( jfn, 8);
                   END;
                ttych(EOL);
                END;
             = rcvdone:
                BEGIN
```

```
                IF debug.dbprim THEN ttyas($"Receive binary
                completed ");
                IF debug.dbparam THEN
                   BEGIN
                   typen(length, 10);
                   ttyas($" bytes from jfn ");
                   typen( jfn, 8);
                   END;
                ttych(EOL);
                IF debug.dbmsg THEN typemessg( &blk);
                END;
             ENDCASE err($"Bad status -TYPEMSG");
      ENDCASE
         BEGIN
         ttyas($"Unknown primitive: "); typen(type, 10);
         ttych(EOL);
         END;
   RETURN;
   END.

(typesig) %type msg-3 signal%
PROCEDURE (pb REF);                                              4C6
   LOCAL STRING sigstring[30];
   ttyas($" sig: ");
   CASE pb[sig].sigtype OF
      = psi:
         *sigstring* _ "psi ", STRING(pb[sig].sigdata, 8), "B;
      = unblock:
         *sigstring* _ "unblock";
      ENDCASE err($"Bad signal in MSG-3 -TYPESIG");
   ttyas($sigstring);
   RETURN;
   END.

(typen) % type out a number %
PROCEDURE ( number, base);                                       4C7
   LOCAL STRING numstring[14];
   CASE base OF
      = 8:
         BEGIN
         IF number.LH NOT= 0 THEN *numstring* _ STRING(number.LH,
         base);
         *numstring* _ *numstring*, STRING(number.RH, base), "B;
         END;
      ENDCASE
         *numstring* _ STRING(number, base);
   ttyas($numstring);
   RETURN;
   END.
```

```
< NINE, ADRMNP.NLS;16, >, 21-Jul-78 13:25 HGL ;;;;
FILE adrmnp  % <ARCSUBSYS>L109  <RELNINE>adrmnp % % (ARCSUBSYS,L109,)
(RELNINE,adrmnp.rel,)  %
   % DOCUMENTATION %
      % the following is the formal description of a link that is used
      by the procedures that parse links:
         s  :=  [SP / TAB / CR / LF / EOL]  /  s (SP / TAB / CR / LF /
         EOL)                                                          1A1A
         link  :=  opndlm  s  body  s  clsdlm                          1A1B
            opndlm  :=  ( ('< / '() [comment] )  /  "--"               1A1B1
            clsdlm  :=  ')  /  '>                                      1A1B2
            comment  :=  ctxt "--"                                     1A1B3
            ctxt  :=  any number of characters excluding the following:
                                                                       1A1B4
               '- / ', / '> / ') / '< / '( / '. / '; / ': /            1A1B4A
               '! / '* / '# / '/ / '\ / '" / '' / '= / '+
         body  :=  [filspc]  [dae]  ['': vwspc]                        1A1C
         filspc  :=  [[hn ',] un ',] fn ',                             1A1D
         hn  :=  s hstnam s  /  NULL                                   1A1E
            hstnam  :=  1$48(LD/'-)                                    1A1E1
         un  :=  s usrnam s                                            1A1F
            usrnam  :=  zero to 39 characters excluding the following
                                                                       1A1F1
               ', / SP / TAB / EOL / ': / '; /
               '< / '> / '= / '_ / '* / '@ / '. /
               [0B,5B] / [7B,32B] / [34B,36B] / 140B / >= 173B
         fn  :=  s (filnam / filnam2) s                                1A1G
            filnam  :=  zero or more characters excluding the following
                                                                       1A1G1
               ', / SP / TAB / EOL / ': /
               '< / '> / '= / '_ / '@ /
               [0B,5B] / [7B,32B] / [34B,36B] / 140B / >= 173B
            filnam2  :=  '= delim1 STRING delim2                       1A1G2
            delim1  :=  CHARACTER                                      1A1G3
            delim2  :=  the same character used for delim              1A1G4
         dae  :=  element  /  dae element                              1A1H
         element  :=                                                   1A1I
            s  /                                                       1A1I1
            stmntnam  /  '! stmntnam  /  '* stmntnam  /                1A1I2
            '# marker  /
            '/  /  '\  /
            ('" STRING '"  /  '' CHAR) s ['= search]  /
            '. pelement  /  '+ selement  /  '- selement
         marker  :=                                                    1A1J
            a letter followed by any number of characters except the
            following:                                                 1A1J1
               SP / TAB / ') / '> / ':                                 1A1J1A
         stmntnam  :=                                                  1A1K
            LD  [LD / '- / '' / '@]  /  stmntnam  (LD / '- / '' / '@)
                                                                       1A1K1
         search  :=  stype [sdomain]  /  sdomain [stype]               1A1L
         stype  :=  [NUMBER] ('C  /  'W)                               1A1M
            (can be upper or lower case letters)
         sdomain  :=  [NUMBER] 'S                                      1A1N
            (can be upper or lower case letters)
         selement  :=  [NUMBER] struc  /  selement [NUMBER] struc      1A1O
```

```
struc  :=  'C  /  'E  /  'F  /  'I  /  'L  /  'N  /  'V  /  'W
                                                              1A1P
   (can be upper or lower case letters)
pelement  :=  [NUMBER] pos  /  pelement [NUMBER] pos          1A1Q
pos  :=                                                       1A1R
   'B  /  'C  /  'D  /  'E  /  "FR"  /  'H  /  'L  /         1A1R1
   'N  /  'O  /  'P  /  'R  /  'S  /  'T  /  'U  /  'W       1A1R2
      (can be upper or lower case letters)
vwspc  :=  [viewspec]  /  vwspc viewspec                      1A1S
viewspec  :=                                                  1A1T
    s  /
   'a  /  'b  /  'c  /  'd  /  'e  /  'f  /  'g  /  'h  / 1A1T2
   'i  /  'j  /  'k  /  'l  /  'm  /  'n  /  'o  /  'p  / 1A1T3
   'q  /  'r  /  's  /  't  /  'u  /  'v  /  'w  /  'x  / 1A1T4
   'y  /  'z  /  'A  /  'B  /  'C  /  'D  /  'G  /  'H  / 1A1T5
   'I  /  'J  /  'K  /  'L  /  'O  /  'P  /  filter       1A1T6
filter  :=  '; CONTENT-PATTERN ';                             1A1U
%
```

```
% Link field extraction. %
   (lnkpspc)  %extract fields from link and setup link data block%
                                                                    1B1
      PROCEDURE
         (lnknum,              %link number%
         tptadr,               %t-ptr from which to start link search%
         usrstg,               %string for directory name%
         fnmstg,               %string for file name%
         addexpstg,  %string for address expression%
         vspstg,               %string for viewspecs%
         adstr);               % addres of data block to be filled %
      LOCAL
         ind,
         rhstn;       % host number %
      LOCAL TEXT POINTER
         tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8;
      REF tptadr, usrstg, fnmstg, addexpstg, vspstg, adstr;

      % find and parse the right link and get the file name %
         IF &tptadr THEN
            BEGIN
            tp8 _ tptadr;  tp8[1] _ tptadr[1];
            ind _ 1;
            DO
               BEGIN
               rhstn _ lnbfls( $tp8, &adstr, &fnmstg);
               CASE ind OF
                  = 1: NULL;
                  ENDCASE
                     IF adstr[ls+1] <= tp8[1] THEN
                        BEGIN
                        ABORT( lnk5err, $"Illegal Link:
                        Left Delimiter Not Found");
                        END;
               tp7 _ adstr[ls];  tp7[1] _ adstr[ls+1];
               tp8 _ adstr[le];  tp8[1] _ adstr[le+1];
               BUMP ind;
               END UNTIL ind > lnknum;
            END
         ELSE
            BEGIN
            rhstn _ lnbfls( 0, &adstr, &fnmstg );
            tp7 _ adstr[ls];  tp7[1] _ adstr[ls+1];
            END;
      % set up other strings %
         tp1 _ adstr[us];  tp1[1] _ adstr[us+1];
         tp2 _ adstr[ue];  tp2[1] _ adstr[ue+1];
         *usrstg* _ + tp1 tp2; % directory %
         tp3 _ adstr[ds];  tp3[1] _ adstr[ds+1];
         tp4 _ adstr[de];  tp4[1] _ adstr[de+1];
         *addexpstg* _ tp3 tp4; % address expression %
         tp5 _ adstr[vb];  tp5[1] _ adstr[vb+1];
         tp6 _ adstr[ve];  tp6[1] _ adstr[ve+1];
         *vspstg* _ tp5 tp6; % view %
      RETURN( rhstn );
      END.
```

% %

```
(lnbfls)  % create file name string from link parse block%     1B2
   PROCEDURE
      (stp,       % FALSE or t-ptr for use for parsing a link %
      adstr,      % address of parsed link data structure or zero
      %
      astr);      % address of string to get file name %
   LOCAL
      i,  % index for copying data structures %
      rhstn,      % cell to get remote host number %
      ldstr[40];  % local data structure for link parsing %
   LOCAL TEXT POINTER
      h1, h2, u1, u2, f1, f2;
   LOCAL STRING
      locstr[200];
   REF stp, adstr, astr;

   % RETURNS %
      % will generate error if invalid hostname field in link or
      call to "gdftdir" fails (invalid directory number).
      Otherwise, will return filled in string that can be used
      for gtjfn and will return hostnumber (number for local host
      if none specified in the link %

   % ALGORITHM %
      % for local hosts the following is how the file string is
      built up:
         if a user name is specified (that doesn't end with a
         control-f or an altmode) then the file name is the
         concatenation of:
            "< specified-user-name "> specified-file-name
               if the file name is not specified, it is set to
               the file currently loaded (or generates an error
               for typed in links)
         if a user name is specified (that ends with a control-f
         or an altmode) then the file name is the concatenation
         of:
            "< specified-user-name specified-file-name
               if the file name is not specified, it is set to
               the file currently loaded (or generates an error
               for typed in links)
         if no user name is specified for a typed in link, then
         the file name string is merely the specified file name
         (or NULL if no file name specified)
         if no user name is specified for a bugged link, then the
         file name string consists of the concatenation of:
            "< default-directory "> specified-file-name
               if the file name is not specified, it is set to
               the file currently loaded
      for remote hosts, if a user name is specified, then the
      same algorithm is used as is used for the local host.
      for remote hosts without a user name, the returned string
      is the specified file name (or generates an error if no
      file name specified for a typed in link)
      %

   % parse link or copy data structure as necessary %
```

```
      IF &stp THEN
         BEGIN
         lnkprs( &stp, $ldstr);
         IF &adstr THEN
            FOR i _ 1 UP UNTIL > lnkdsl DO adstr[i-1] _
            ldstr[i-1];
         END
      ELSE
         IF &adstr THEN
            FOR i _ 1 UP UNTIL > lnkdsl DO ldstr[i-1] _
            adstr[i-1]
         ELSE err( $"NLS System Error: Illegal Call to LNBFLS");
   % initialize text pointers %
      h1 _ ldstr[hs];  h1[1] _ ldstr[hs+1];
      h2 _ ldstr[he];  h2[1] _ ldstr[he+1];
      u1 _ ldstr[us];  u1[1] _ ldstr[us+1];
      u2 _ ldstr[ue];  u2[1] _ ldstr[ue+1];
      f1 _ ldstr[fs];  f1[1] _ ldstr[fs+1];
      f2 _ ldstr[fe];  f2[1] _ ldstr[fe+1];
   % pick up any specified user name %
      *locstr* _ + u1 u2;
   % act according to whether its a local host or not %
      CASE (rhstn _ gthstr( $h1, $h2 ) ) OF
         < 0:  err($"invalid host name");
         = lhostn:        % local hosts %
            (lcl):                                          1B2N1B1
            CASE locstr.L OF
               = 0:       % no user name specified %
                  BEGIN
                  *locstr* _ f1 f2;
                  CASE ldstr[lfn] OF
                     = 0: % typed in link %
                        IF locstr.L THEN *astr* _ *locstr*
                        ELSE *astr* _ NULL;
                     ENDCASE      % bugged link %
                        BEGIN
                        IF NOT gdftdir( ldstr[lfn], &astr) THEN
                           BEGIN %pass it if it's a journal item%
                           IF FIND f1 D THEN *astr* _ NULL
                           ELSE err ($"illegal username in file
                           header");
                           END
                        ELSE *astr* _ '<, *astr*, '>;
                        IF locstr.L THEN *astr* _ *astr*,
                        *locstr*
                        ELSE lnbafn( &astr, ldstr[lfn] );
                        END;
                  END;
               ENDCASE    % user name specified %
                  BEGIN
                  CASE *locstr*[locstr.L] OF
                     = '<^F>, = '<ESC>:
                        *astr* _ '<, *locstr*;
                     ENDCASE
                        *astr* _ '<, *locstr*, '>;
                  *locstr* _ f1 f2;
```

```
                    CASE locstr.L OF
                       = 0:
                          IF ldstr[lfn] THEN lnbafn( &astr,
                          ldstr[lfn] )
                          ELSE err($"Illegal Link");
                       ENDCASE
                          *astr* _ *astr*, *locstr*;
                    END;
          ENDCASE  % remote hosts %
             IF locstr.L THEN GOTO lcl
             ELSE
                BEGIN
                *astr* _ f1 f2;
                IF NOT astr.L THEN
                   IF NOT ldstr[lfn] THEN err($"Illegal Link")
                   ELSE
                      lnbafn( &astr, ldstr[lfn] );
                END;
       % now return %
          RETURN( rhstn );
       END.

       %  %
```

```
(inbafn)  % append file name string to passed string %          1B3
   PROCEDURE
      (astr,       % string to be appended to %
      fileno);     % file number for file name string %
   LOCAL
      ldstr[40];  % data structure to parse link %
   LOCAL TEXT POINTER
      tp1, tp2, tp3;
   LOCAL STRING locstr[200];
   REF astr;

   % get the file name %
      filnam( fileno, $locstr);
   % now parse the link %
      FIND SF(*locstr*) ^tp3;
      lnkprs( $tp3, $ldstr);
      tp1 _ ldstr[fs];  tp1[1] _ ldstr[fs+1];
      tp2 _ ldstr[fe];  tp2[1] _ ldstr[fe+1];
   % do the appending and return %
      *astr* _ *astr*, tp1 tp2;
      RETURN;
   END.

   %  %
```

```
(lnkprs)  % detailed link parser %                                    1B4
   PROCEDURE
      (stp,        % address of text pointer to start scan %
      adstr        % address of  linkparams  data strcture %
      );
   LOCAL rflag;
   LOCAL TEXT POINTER tp1, tp2;
   REF stp, adstr;

   % the following signals (maintained in CONST) can be generated
   while parsing a link:
      lnk1err -
         Illegal Link Syntax:
         ENDCHR Before Closing "
      lnk2err -
         Illegal Link Syntax:
         ENDCHR Before CH After '
      lnk3err -
         Illegal Link Syntax:
         Illegal Statement Name or Number After ! or *
      lnk4err -
         Illegal Link Syntax:
         Illegal DAE Element
      lnk5err -
         Illegal Link:
         Left Delimeter Not Found
      lnk6err -
         Illegal Link Syntax or Semantic:
         Missing Right Delimeter or Bad Viewspecs
      lnk7err -
         Illegal Link Syntax:
         Missing Right Delimeter
      lnk8err -
         Illegal Link Syntax:
         Illegal Marker After #
      lnk9err -
         Illegal Link Syntax:
         ENDCHR Before ; In Filter
      %

   % initialize %
      rflag _ TRUE;
   % setup working t-ptr %
      FIND stp ^tp1 ^tp2;
      IF FIND > '- THEN FIND < $'- ^tp1 ^tp2;
   % setup default directory file number %
      adstr[lfn] _ stp.stfile;
   % try different starting points if errors %
      INVOKE (signot5, lnkps1);
   (lnkps1):                                                         1B4L
   % find start of the link %
      lnkstr( $tp1, &adstr);
   % find an optional comment %
      lnkcom( &adstr );
   % find the body of the link %
      lnkbdy( &adstr );
```

```
    % find the right delimeter %
       lnkend( &adstr );
    % done, so return %
       DROP (signot5);
       RETURN;

    (signot5) CATCHPHRASE;                                          1B4S
       BEGIN
       DISABLE (signot5);
       CASE SIGNAL OF
          # lnk5err:
             IF rflag AND (adstr[ls+1] >= tp1[1]) THEN
                BEGIN
                tp1[1] _ adstr[cs+1];
                TERMINATE; %GOTO lnkps1%
                END
             ELSE
                BEGIN
                rflag _ FALSE;
                tp1[1] _ MIN(tp1[1], tp2[1], adstr[ls+1]);
                IF FIND tp1 < [".( / "< / "--"] ^tp1 >
                   THEN TERMINATE;
                END;
          ENDCASE
             CONTINUE;
          END;

    END.

    %  %
```

```
(lnkstr) %***%    % find the start of a link %                    185
   PROCEDURE
      (stp,       % address of text pointer to start scanninfg
      from %
      adstr       % address of  linkparams  data strcture %
      );
   LOCAL lnktyp;
   LOCAL TEXT POINTER tp1;
   REF stp, adstr;

   % ABNORMAL RETURNS %
      % can generate the following signal:
         lnk5err:
            Illegal Link:
            Left Delimeter Not Found
      %

   % position to scan starting text pointer %
      FIND stp > ^tp1;
      IF NOT stp.stastr THEN
         BEGIN
         tp1[1] _ MAX( tp1[1], fchtxt(stp) );
         FIND tp1 >;
         END;
   % initially normal type link %
      lnktyp _ FALSE;
   LOOP
      CASE READC OF
         = "(, = "<:
            BEGIN
            FIND ^tp1 _tp1;
            EXIT LOOP;
            END;
         = "-:
            IF FIND 1$"- < 2CH ^tp1 THEN
               BEGIN
               lnktyp _ TRUE;
               EXIT LOOP;
               END;
         = ENDCHR, = "), = ">:
            BEGIN
            FIND stp <;
            LOOP
               CASE READC OF
                  = "(, = "<:
                     BEGIN
                     FIND ^tp1;
                     EXIT LOOP 2;
                     END;
                  = "-:
                     IF FIND "- ^tp1 THEN
                        BEGIN
                        lnktyp _ TRUE;
                        EXIT LOOP 2;
                        END;
                  = ENDCHR:
```

```
                    BEGIN
                    tp1[1] _ 0;
                    EXIT LOOP 2;
                    END;
                ENDCASE;
            END;
        ENDCASE;
IF (NOT tp1[1]) OR (NOT stp.stastr) THEN
   IF (NOT tp1[1]) OR (tp1[1]<fchtxt(stp)) THEN
      BEGIN
      ABORT( lnk5err, $"Illegal Link:
      Left Delimiter Not Found");
      END;
adstr[ls] _ tp1;  adstr[ls+1] _ tp1[1];
% initialize to no comment %
   adstr[cs] _ adstr[ce] _ adstr[ls];
   adstr[cs+1] _ adstr[ce+1] _ adstr[ls+1]+1;
% no comment if link starts with "--" %
   IF lnktyp THEN
      BEGIN
      % make comment live after "--" %
         adstr[cs+1] _ adstr[ce+1] _ adstr[ls+1] + 2;
      END;
RETURN;
END.

%  %
```

```
(lnkcom)  % find the optional comment in a link %                    186
   PROCEDURE
      (adstr       % address of  linkparams  data strcture %
      );
   LOCAL TEXT POINTER tp1;
   REF adstr;

   % initialize %
      tp1 _ adstr[cs];
      tp1[1] _ adstr[cs+1];
   % no comment if link starts with "--" %
      IF (adstr[ce+1] - adstr[cs+1]) = 2 THEN RETURN;
   % setup to scan forward from start of link looking for comment
   %
      FIND tp1 >;
   % scan until "--" or any other link delimeter %
      LOOP
         CASE READC OF
            = "-:
               IF FIND 1$"- ^tp1 THEN
                  BEGIN
                  adstr[ce] _ tp1;
                  adstr[ce+1] _ tp1[1];
                  RETURN;
                  END
               ELSE RETURN;
            = ENDCHR,
            = ",, = ">, = "), = "<, = "(, = ".,
            = ";, = ":, = "!, = "*, = "#, = "/,
            = "\, = "", = "", = "=, = "+:
               RETURN;
            ENDCASE;

   END.

   %  %
```

```
(lnkbdy)  % find the body of a link %                                  187
   PROCEDURE
      (adstr      % address of  linkparams  data strcture %
      );
   LOCAL TEXT POINTER stp;
   REF adstr;

   % find the optional filspc part of the link %
      lnkfpc( &adstr, $stp );
   % find the optional filspc part of the link %
      lnkdae( &adstr, $stp );
   % find the optional vwspc part of the link %
      lnkvws( &adstr, $stp );
   % done, so return %
      RETURN;

   END.

   %  %
```

```
(lnkfpc)  % find the optional filspc of a link %                    188
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %
      tpl % address of text pointer to get end of filspc %
      );
   REF adstr, tpl;

   % find an optional hostname %
      lnkhst( &adstr, &tpl );
   % find an optional username %
      lnkusr( &adstr, &tpl );
   % find an optional filename %
      lnkfil( &adstr, &tpl );
   % done so return %
      RETURN;

   END.

   %  %
```

```
(lnkhst)  % find the optional hostname in a link %                    1B9
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %
      stp % text pointer gets end of hostname field %
      );
   LOCAL TEXT POINTER tp1, tp2;
   REF adstr, stp;

   % initialize to no hostname %
      adstr[hs] _ adstr[he] _ tp1 _ adstr[ce];
      adstr[hs+1] _ adstr[he+1] _ tp1[1] _ adstr[ce+1];
   % setup scan forward from end of comment (skip spaces or tabs)
   %
      FIND tp1 > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;
   % hostname is 0-48 letters, digits, or minus signs, followed
   by any number of spaces or tabs, and terminated by a comma %
      IF FIND $48(LD/'-) ^tp2 $(SP/TAB/CR/LF/EOL) ', ^stp THEN
         BEGIN
         adstr[hs] _ tp1;
         adstr[hs+1] _ tp1[1];
         adstr[he] _ tp2;
         adstr[he+1] _ tp2[1];
         RETURN;
         END
      ELSE RETURN;

   END.

   %  %
```

```
(lnkusr)  % find the optional username in a link %                1B10
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %
      stp % text ptr to start scan; gets end of username field %
      );
   LOCAL
      i    % character count index %
      ;
   LOCAL TEXT POINTER tp1, tp2;
   REF adstr, stp;

   % initialize to no username %
      adstr[us] _ adstr[ue] _ adstr[he];
      adstr[us+1] _ adstr[ue+1] _ adstr[he+1];
   % setup scan forward from end of hostname (skip spaces & tabs)
   %
      FIND stp > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;
   % find username if it exists, else make hostname username and
   make hostname null %
      FOR i _ 1 UP UNTIL > 39 DO
         CASE READC OF
            % illegal username characters %
               = ",, = SP, = TAB, = CR, = LF, = EOL:
                  BEGIN
                  IF NOT FIND < CH ^tp2 > $(SP/TAB/CR/LF/EOL) ",
                  ^stp THEN
                     GOTO notusr;
                  adstr[us] _ tp1;  adstr[us+1] _ tp1[1];
                  adstr[ue] _ tp2;  adstr[ue+1] _ tp2[1];
                  RETURN;
                  END;
               = ":, = ";, = "<, = ">, = "=, = "_, = "*, = "@, =
               ".,
               = ENDCHR,
               IN [0B,5B] % ^@ - ^E %,
               IN [7B,32B]        % ^G - ^Z %,
               IN [34B,36B]       % ^] - ^^ %,
               = 140B, >= 173B:
                  (notusr):       BEGIN                    1B10H1A1G1
                  % make username field = hostname field %
                     adstr[us] _ adstr[hs];
                     adstr[us+1] _ adstr[hs+1];
                     adstr[ue] _ adstr[he];
                     adstr[ue+1] _ adstr[he+1];
                  % make hostname field null %
                     adstr[he] _ adstr[hs];
                     adstr[he+1] _ adstr[hs+1];
                  RETURN;
                  END;
            ENDCASE;
      % more than 39 characters is not a user name %
         GOTO notusr;

   END.

   %  %
```

```
(lnkfil)  % find the optional filename in a link %                1B11
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %
      stp % text ptr to start scan; gets ent of filename field %
      );
   LOCAL
      char        % delimeter character %
      ;
   LOCAL TEXT POINTER tp1, tp2;
   REF adstr, stp;

   % initialize to no filename %
      adstr[fs] _ adstr[fe] _ adstr[ue];
      adstr[fs+1] _ adstr[fe+1] _ adstr[ue+1];
   % setup scan forward from end of username (skip spaces & tabs)
   %
      FIND stp > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;
   % find filename if it exists, else make username filename and
   make username hostname and make hostname null %
      CASE READC OF
         = "=;
            CASE char _ READC OF
               = ENDCHR:  GOTO notfil;
               ENDCASE
                  BEGIN
                  FIND ^tp1;
                  LOOP
                     CASE READC OF
                        = ENDCHR:  GOTO notfil;
                        = char:
                           IF FIND < CH ^tp2 > CH
                           $(SP/TAB/CR/LF/EOL) ", ^stp
                              THEN GOTO isfil
                           ELSE GOTO notfil;
                        ENDCASE;
                  END;
         = ENDCHR: GOTO notfil;
         ENDCASE
            BEGIN
            FIND < CH >;
            LOOP
               CASE READC OF
                  % illegal filename characters %
                     = ",, = SP, = TAB, = CR, = LF, = EOL:
                        BEGIN
                        IF NOT FIND < CH ^tp2 >
                        $(SP/TAB/CR/LF/EOL) ", ^stp
                           THEN GOTO notfil;
                        (isfil):                          1B11H1C3A1A3
                           adstr[fs] _ tp1;  adstr[fs+1] _
                           tp1[1];
                           adstr[fe] _ tp2;  adstr[fe+1] _
                           tp2[1];
                           RETURN;
                        END;
                     = ":, = "<, = ">, = "=, = "_, = "@,
```

```
                    = ENDCHR,
                    IN [0B,5B]   % ^@ - ^E %,
                    IN [7B,25B]  % ^G - ^U %,
                    IN [27B,32B] % ^W - ^Z %,
                    IN [34B,36B] % ^] - ^^ %,
                    = 140B, >= 173B:
                       (notfil): BEGIN                    1B11H1C3A1H1
                       % make filename field = username field %
                          adstr[fs] _ adstr[us];
                          adstr[fs+1] _ adstr[us+1];
                          adstr[fe] _ adstr[ue];
                          adstr[fe+1] _ adstr[ue+1];
                       % make username field = hostname field %
                          adstr[us] _ adstr[hs];
                          adstr[us+1] _ adstr[hs+1];
                          adstr[ue] _ adstr[he];
                          adstr[ue+1] _ adstr[he+1];
                       % make hostname field null %
                          adstr[he] _ adstr[hs];
                          adstr[he+1] _ adstr[hs+1];
                       RETURN;
                       END;
                 ENDCASE;
           END;

     END.

     % %
```

```
(lnkdae)  % find the optional dae in a link %                    1B12
   PROCEDURE
      (adstr,     % address of  linkparams  data strcture %
      stp % text pointer to start scan; gets end of dae %
      );
   LOCAL
      lnkparE29]  % additionl link params block for recursion %
      ;
   LOCAL TEXT POINTER tp1, tp2, tp3;
   REF adstr, stp;

   % ABNORMAL RETURNS %
      % this procedure can generate the following signals:
         lnk2err:
            Illegal Link Syntax:
            ENDCHR Before CH After "
      %

   % initialize to no dae %
      (daeprs):                                                  1B12H1
      adstrEds] _ adstrEde] _ adstrEfe];
      adstrEds+1] _ adstrEde+1] _ adstrEfe+1];
   % setup scan forward from end of filename (skip spaces & tabs)
   %
      FIND stp > $(SP/TAB/CR/LF/EOL) ^tp1 ^stp;
   % find dae if it exists %
      LOOP
         CASE READC OF
            = ":
               BEGIN
               FIND ^stp < CH $(SP/TAB/CR/LF/EOL) ^tp2;
               (lkgdae):                                     1B12J1A1C
                  IF tp2E1] < tp1E1] THEN tp2E1] _ tp1E1];
                  adstrEds] _ tp1;  adstrEds+1] _ tp1E1];
                  adstrEde] _ tp2;  adstrEde+1] _ tp2E1];
                  RETURN;
               END;
            = "), = ">:
               BEGIN
               FIND < CH $(SP/TAB/CR/LF/EOL) ^tp2 ^stp;
               GOTO lkgdae;
               END;
            = "":
               LOOP
                  CASE READC OF
                     = "", = ENDCHR:  EXIT LOOP;
                     ENDCASE;
            = "":
               CASE READC OF
                  = ENDCHR:
                     BEGIN
                     ABORT( lnk2err, $"Illegal Link Syntax:
                     ENDCHR Before CH After "");
                     END;
                  ENDCASE;
            = "(, = "<, ="",, = ENDCHR: % some common bad chrs %
```

```
            IF adstr[fe+1] > adstr[fs+1] THEN  % backup %
               BEGIN
               % make filename field = username field %
                  adstr[fs] _ adstr[us];
                  adstr[fs+1] _ adstr[us+1];
                  stp _ adstr[fe] _ adstr[ue];
                  stp[1] _ adstr[fe+1] _ adstr[ue+1];
               % make username field = hostname field %
                  adstr[us] _ adstr[hs];
                  adstr[us+1] _ adstr[hs+1];
                  adstr[ue] _ adstr[he];
                  adstr[ue+1] _ adstr[he+1];
               % make hostname field null %
                  adstr[he] _ adstr[hs];
                  adstr[he+1] _ adstr[hs+1];
               % now reparse the dae %
                  FIND stp > $(SP/TAB/CR/LF/EOL) ', ^stp;
                  GOTO daeprs;
               END
            ELSE RETURN;
         ENDCASE;

END.

%  %
```

```
(lnkvws)  % find the optional vwspc in a link %                    1B13
   PROCEDURE
      (adstr,      % address of  linkparams  data strcture %
      stp % text pointer to start scan; gets end of dae %
      );
   LOCAL TEXT POINTER tp1, tp2;
   REF adstr, stp;

   % ABNORMAL RETURNS %
      % this routine can generate the following signal:
         lnk6err:
            Illegal Link Syntax or Semantic:
            Missing Right Delimeter or Bad Viewspecs
         lnk9err:
            Illegal Link Syntax:
            ENDCHR Before ; In Filter
      %

   % initialize to no vwspc %
      adstr[vb] _ adstr[ve] _ adstr[de];
      adstr[vb+1] _ adstr[ve+1] _ adstr[de+1];
   % setup scan forward from end of dae (skip spaces & tabs) %
      FIND stp > $(SP/TAB) ^tp1 ^stp;
   % find vwspc if it exists %
      LOOP
         CASE READC OF
            =SP, =TAB, =CR, =LF, =EOL, IN["a,"z], IN["A,"Z]:
               NULL;
            = "), =">, = ENDCHR:
               BEGIN
               FIND < CH $(SP/TAB/CR/LF/EOL) ^tp2;
               IF tp2[1] < tp1[1] THEN tp2[1] _ tp1[1];
               adstr[vb] _ tp1;  adstr[vb+1] _ tp1[1];
               adstr[ve] _ tp2;  adstr[ve+1] _ tp2[1];
               RETURN;
               END;
            = ";:
               LOOP
                  CASE READC OF
                     = ";:  EXIT LOOP;
                     = "":
                        READC;
                     = "":
                        LOOP
                           CASE READC OF
                              = "": EXIT LOOP;
                              = ENDCHR: EXIT LOOP;
                              ENDCASE;
                     = ENDCHR:
                        BEGIN
                        ABORT( lnk9err, $"Illegal Link Syntax:
                        ENDCHR Before ; In Filter");
                        END;
                     ENDCASE;
            ENDCASE
               BEGIN
```

```
            ABORT( lnk6err, $"Illegal Link Syntax or
            Semantics:
            Missing Right Delimiter or Bad Viewspecs");
            END;

    END.

    %  %
```

```
(lnkend)  % find the right delimeter of a link %                    1814
   PROCEDURE
      (adstr      % address of  linkparams  data strcture %
      );
   LOCAL TEXT POINTER tp1;
   REF adstr;

   % ABNORMAL RETURNS %
      % this routine can generate the following signals:
         lnk7err:
            Illegal Link Syntax:
            Missing Right Delimeter
      %

   % setup to scan from end of viewspecs %
      tp1 _ adstr[ve];  tp1[1] _ adstr[ve+1];
   % find the end of the link %
      IF NOT FIND tp1 > $(SP/TAB/CR/LF/EOL) (")/">) ^tp1 THEN
         BEGIN
         ABORT( lnk7err, $"Illegal Link Syntax:
         Missing Right Delimiter");
         END;
   % now set up the text pointer in the linkparams block %
      adstr[le] _ tp1;  adstr[le+1] _ tp1[1];
   % done, so return %
      RETURN;

   END.

   %  %
```

```
%*************************************************
        Address Expression evaluation routines
*************************************************%
   (caddexp)   % G: compute address expression %
   PROCEDURE (tp1 REF, tp2 REF, da REF, curptr REF % => see RESULTS
   %);                                                          1C1
      % Procedure description
        FUNCTION
           Computes the stid corresponding to the address
           expression passed to it.
        ARGUMENTS
           tp1, tp2 -- TEXT POINTER -- embracing the text of the
           address expression
           da -- display area. needed for viewspecs, file/stmt
           return rings.
              NOTE: THE da WILL ALWAYS STAY IN TACT AND WILL NOT
              CHANGE WHEN PROCEDURE RETURNS.
           curptr -- TEXT POINTER -- reference point for address
           expression.
        RESULTS
           On successful address evaluation returns:
              vs1, vs2 -- the two new viewspec words.
              ca -- address for the new content-pattern or 0;
              sg -- address of the new seq-generator or 0
              srr -- entry of last stmnt return ring referenced or
              0
        NON-STANDARD CONTROL
           Creates signal of type gaderr if anything goes wrong.
           Calls err if access to remote files is attempted
        GLOBALS
           none
        %
      % Declarations %
        LOCAL adstr[40], savvs1, savvs2, savca, savsg;
        LOCAL TEXT POINTER ltp1, ltp2;
        LOCAL STRING loclink[149], locstr[149], dir[39], flnam[99],
        adexp[99], vsstr[49];
      % save da parameters %
        savvs1 _ da.davspec;
        savvs2 _ da.davspc2;
        savca _ da.dacacode;
        savsg _ da.dausgcode;
      % protect agairs errors %
        INVOKE(caderr);
      % copy link to local string %
        IF NOT FIND tp1 > $NP ("(/"<) THEN *loclink* _"<;
        *loclink* _ *loclink*, tp1 tp2, ">;
      % parse the link %
        FIND SF(*loclink*) ^ltp1;
        lnkprs($ltp1, $adstr);
        adstr[lfn] _ tp1.stfile;  %link might have been bugged %
        IF lnkpspc(1, 0, $dir, $flnam, $adexp, $vsstr, $adstr) #
        lhostn THEN
           err( noremote );
      % is a new file involved %
        IF adstr[fe+1] > adstr[fs+1] THEN
```

```
        curptr _ fstid($flnam : curptr[1]);
    % do the in-file address %
      IF adexp.L THEN
        BEGIN
          IF NOT cifadr($adexp, &da, &curptr, $locstr) THEN
            adxerr($locstr);
        END;
    % incorporate the viewspecs %
      IF vsstr.L THEN feedlt(&da, $vsstr);
    % drop the catchphrase %
      DROP(caderr);
    % Return %
      RETURN(
        da.davspec := savvs1,
        da.davspc2 := savvs2,
        da.dacacode := savca,
        da.dausqcode := savsg,
        fakesrr);
    % cathcphrase definition %
      (caderr) CATCHPHRASE();                                    1C1L1
        BEGIN
        CASE SIGNALTYPE OF
          = aborttype :
            BEGIN
              DISABLE( caderr );
              da.davspec _ savvs1;
              da.davspc2 _ savvs2;
              da.dacacode _ savca;
              da.dausqcode _ savsg;
            END;
          ENDCASE;
        CONTINUE;
        END;
    END.

(cifadr)   % U: compute in-file address %
PROCEDURE (adexp REF, da REF, curptr REF, errstr REF % => T/F %);
                                                                 1C2
    % Procedure description
      FUNCTION
        compute in-file address
      ARGUMENTS
        adexp - STRING - in-file address expression
        da - REF - display area (needed for .l constructs only)
        curptr - TEXT POINTER - reference point for addresss
        evaluation
        errstr - STRING - error string in case of failure.
      RESULTS
        TRUE if successful with curptr and da updated.  FALSE
        otherwise.
      NON-STANDARD CONTROL
        Creates a signal when .l construct is used and
        referenced link does not resolve to a stid.
      GLOBALS
        none
      %
```

```
% Declarations %
   LOCAL count, domain,found, char;
   LOCAL TEXT POINTER ltp1;
   LOCAL STRING locadrs[199], locstr[199];
% initialization %
   FIND SF(*adexp*) ^ltp1;
   fakesrr _ FALSE;
LOOP BEGIN % as long as you can %
   % get the next char %
      found _ FALSE;
      FIND ltp1 > $NP (CH FS found/ENDCHR) ^ltp1;
      char _ IF found THEN *adexp*[ltp1[1]-1] ELSE ENDCHR;
   CASE char OF
      = '!, % branch name %
      = '*, % next name %
      = '$, % external name %
      = LD: % statement name/number/sid %
         BEGIN
         BUMP DOWN ltp1[1];  % backup character %
         IF NOT castname($ltp1, &curptr, &errstr) THEN
         RETURN(FALSE);
         END;
      = '/: % slash%
         BEGIN
            ccurcon(&curptr, $locstr);
            dismes(1, $locstr);
         END;
      = '\: % backslash %
         BEGIN
            cprint(&da, curptr, curptr, stmtv, cortn, 0);
         END;
      = '.: % structure element %
         IF NOT caselement($ltp1, &curptr, &da) THEN
            BEGIN
               *errstr* _ *adexp*[ltp1[1]];
               RETURN(FALSE);
            END;
      = '+: % position relation - forwards %
         IF NOT capelement($ltp1, &curptr, 1) THEN
            BEGIN
               *errstr* _ *adexp*[ltp1[1]];
               RETURN(FALSE);
            END;
      = '-: % position relation - backwards %
         IF NOT capelement($ltp1, &curptr, -1) THEN
            BEGIN
               *errstr* _ *adexp*[ltp1[1]];
               RETURN(FALSE);
            END;
      = '": % content search %
         BEGIN
            *locstr* _ NULL;
            gdlit2($locstr, '");
            FIND ^ltp1; % just to update the pointer %
            srchtype _ conspt($ltp1 : count, domain);
            IF NOT consearch($locstr, &curptr, srchtype,
```

```
                count, domain) THEN
                   BEGIN
                      *errstr* _ "", *locstr*, "";
                      RETURN(FALSE);
                   END;
             END;
          = "": % character search %
             BEGIN
                *locstr* _ READC;
                FIND ^ltp1; % just to update the pointer %
                srchtype _ conspt($ltp1 : count, domain);
                IF NOT consearch($locstr, &curptr, srchtype,
                count, domain) THEN
                   BEGIN
                      *errstr* _ "", *locstr*, "";
                      RETURN(FALSE);
                   END;
             END;
          = "#: % marker %
             BEGIN
             *locstr* _ NULL;
             gdlit2($locstr, SP);
             FIND ^ltp1; % update the pointer %
             IF (curptr _ lkmkr($locstr, curptr.stfile:curptr[13]))
             = endfile
             THEN
                BEGIN
                   *errstr* _ *locstr*;
                   RETURN(FALSE);
                END;
             END;
          = ENDCHR: RETURN(TRUE);
          ENDCASE
             BEGIN
                *errstr* _ char;
                RETURN(FALSE);
             END;
    END; %of main loop %
    END.

(castnam)   % U: compute stid of statement name/number %
PROCEDURE (tp REF, curptr REF, name REF % => T/F %);          1C3
    % Procedure description
       FUNCTION
          compute stid of statement name/number. May be preceded
          by "$, "!, "*.
       ARGUMENTS
          tp - TEXT POINTER where to start scanning.
             WILL BE UPDATED AT END.
          curptr - TEXT POINTER - reference point fo address
          evaluation.
          name - STRING - will return here the name it attempts to
          find (can be used as error string on failure)
       RESULTS
          TRUE if successful with curptr updated. FALSE otherwise.
       NON-STANDARD CONTROL
```

```
      creates a signal on external name with illegal external
      link.
   GLOBALS
      none
   %
% Declarations %
   LOCAL type, val;
   LOCAL TEXT POINTER ltp1, ltp2;
   LOCAL STRING locstr[99], eflnm[99];
% initialize %
   FIND tp ^ltp1;
% find the type of name (external, branch, etc) %
   type _ CASE READC OF
      = "$: extname;
      = "*: segname;
      = "!: braname;
      ENDCASE nametyp;
      FIND ^ltp1;
% scan for name delimiters %
   nmdr($ltp1, $ltp1, $ltp2);
% set up parameters %
   FIND ltp2 ^tp;
   curptr[1] _ 1;
   *name* _ ltp1 ltp2;
% return origin if no name %
   IF name.L = 0 THEN
      BEGIN
         curptr.stpsid _ origin;
         RETURN(TRUE);
      END;
% find what to do with name %
   CASE *name*[1] OF
      = "0:   % sid %
         IF (val _ VALUE($name)) = 0 THEN curptr.stpsid _
         origin
         ELSE lookup(&curptr, val, sid);
      IN ["1, "9]:  % statement number %
         curptr _ fechux($name, curptr.stfile);
         ENDCASE % statement name %
            lookup( &curptr, $name, type );
% check for external names %
   IF curptr=endfile AND type=extname THEN
      BEGIN
         % do we have an external file name? %
            IF NOT getenf(curptr.stfile, TRUE, $eflnm) THEN
               RETURN(FALSE);
         % get the origin of the external file %
            FIND SF(*eflnm*) ^ltp1;
            INVOKE(efncatch, efnrtn);
            IF lnbfls( $ltp1, 0, $locstr) # lhostn THEN
               err(noremote);
            curptr _ fstid($locstr : curptr[1]);
         % follow the name %
            *eflnm* _ *name*, ".1";
            IF NOT cifadr($eflnm, tda, &curptr, $locstr) THEN
            curptr _ endfile;
```

```
            DROP(efncatch);
         END;
      % Return % (efnrtn):
         RETURN( curptr#endfile );
      % catchphrase definition %
         (efncatch) CATCHPHRASE( : val);                              1C3K1
            BEGIN
            CASE SIGNALTYPE OF
               = aborttype :
                  BEGIN
                     curptr _ endfile;
                     TERMINATE;
                  END;
               ENDCASE;
            CONTINUE;
            END;
      END.

   (caselement)   % L: compute s-element expression %
   PROCEDURE (tp REF, curptr REF, da REF % TRUE/FALSE %);               1C4
      % Procedure description
         FUNCTION
            evaluate structure related expression
         ARGUMENTS
            tp - TEXT POINTER - Where to start scanning the
            expression.
               WILL BE UPDATED.
            curptr - TEXT POINTER - reference point for address
            evaluation
            da -- display area (needed for .l constructs)
         RESULTS
            TRUE for success FALSE for failure
               For failure tp will point to the bad character
         NON-STANDARD CONTROL
            creates a signal on the following cases:
               unsuccessful content/word search
               illegal statement return ring (in .fr and .r
               construct)
               a variety of errors for the .l construct
         GLOBALS
            none
         %
      % Declarations %
         LOCAL sring, proc, char, count, param1, frname REF,
         adstr[40];
         LOCAL TEXT POINTER ltp1, ltp2;
         LOCAL STRING str[99];
      LOOP % until relative expression exhausted %
         BEGIN
            % prepare the arguments and defaults %
               proc _ TRUE;
               count _ gadnum(&tp);
               char _ READC;
               FIND ^tp;  % update the pointer %
            % check the cases %
               CASE char OF
```

```
= 'B, ='b: % back %
   param1 _ cwback;
= 'C, = 'c: % content %
   BEGIN
      IF NOT consearch($conreg, &curptr, contnt,
      count, -1) THEN adxerr($conreg);
      proc _ FALSE;
   END;
= 'D, = 'd: % down %
   param1 _ cwdown;
= 'E, = 'e: % end %
   param1 _ cwend;
= 'F, = 'f: % file return %
   CASE char _ READC OF
      = 'R, = 'r:
         BEGIN
            FIND ^tp; % update the pointer %
            proc _ FALSE;
            INVOKE (badfrr);
            &frname _ readfrring(da.dalink, count
            : fakesrr);
            DROP (badfrr);
            FIND SF(*frname*) ^ltp1;
            IF lnbfls($ltp1, 0, $str) # lhostn
            THEN err(noremote);
            curptr _ readsrring(fakesrr, 0 :
            curptr[1], da.davspec, da.davspc2);
            curptr.stfile _ cloafil($str);
         END;
      ENDCASE % error %
         RETURN(FALSE);
= 'H, = 'h: % head %
   param1 _ cwhead;
= 'L, = 'l: % indirect link %
   BEGIN
   curptr[1] _ MAX(fchtxt(curptr), curptr[1]);
   IF lnkpspc(
      count, &curptr, $str, $str, $str, $str,
      $adstr) # lhostn
   THEN err(noremote);
   da.davspec _ caddexp( $adstr[ls], $adstr[le],
   &da, &curptr : da.davspc2, da.dacacode,
   da.dausqcod );
   proc _ FALSE;
   END;
= 'N, = 'n: % next %
   param1 _ cwnext;
= 'O, = 'o: % origin %
   BEGIN
      curptr.stpsid _ origin;
      curptr[1] _ 1;
      proc _ FALSE;
   END;
= 'P, = 'p: % predecessor %
   param1 _ cwpredecessor;
= 'R, = 'r: % return %
```

```
                BEGIN
                   IF NOT fakesrr THEN
                      BEGIN
                         INVOKE (badfrr);
                         readfrring(da.dalink, 0 : fakesrr);
                         DROP (badfrr);
                      END;
                   curptr _ readsrring(fakesrr, count :
                   curptr[1], da.davspec, da.davspc2);
                   proc _ FALSE;
                END;
             = 'S, = 's: % successor %
                param1 _ cwsuccessor;
             = 'T, = 't: % tail %
                param1 _ cwtail;
             = 'U, = 'u: % up %
                param1 _ cwup;
             = 'W, = 'w: % word %
                BEGIN
                   IF NOT consearch($conreg, &curptr, wordtyp,
                   count, -1) THEN adxerr($conreg);
                   proc _ FALSE;
                END;
             = ENDCHR: % end of expression %
                RETURN( TRUE );
             = LD:
                BEGIN
                   FIND tp < CH > ^tp; % update the pointer %
                   RETURN( FALSE );
                END;
             ENDCASE
                RETURN( FIND tp < CH > ^tp ); % will be true %
          % call the procedure if necessary %
             IF proc THEN getpr(param1, count, &curptr);
       END; % of main loop %
    % catchphrase definition %
       (badfrr) CATCHPHRASE();                                    1C4D1
          BEGIN
          DISABLE (badfrr);
          CASE SIGNALTYPE OF
             = aborttype :
                IF SIGNAL = frremptyentry THEN
                   err($"No current entry in file return ring.");
             ENDCASE;
          CONTINUE;
          END;
    END.

(capelemet)   % L: compute adress of position element %
PROCEDURE (tp REF, curptr REF, direction % => T/F %);          1C5
    % Procedure description
       FUNCTION
          compute adress of position element
       ARGUMENTS
          tp -TEXT POINTER - where to start scanning the
          expression
```

```
            WILL BE UPDATED!
         curptr - TEXT POINTER - reference point for address
         evaluation
         direction - INTEGER - +1 forwards, -1 backwards.
      RESULTS
         TRUE for success (with tp and curptr updated) FALSE
         otherwise.
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL result, count, char, scnbck;
      LOCAL TEXT POINTER tp1, tp2, pscan;
   % initialize %
      FIND tp > $NP ^tp;
      result _ FALSE;
   LOOP BEGIN % as long as you can %
      % prepare the arguments %
         count _ gadnum(&tp)*direction;
         char _ READC;
         IF (scnbck _ (count < 0)) THEN
            BEGIN
            FIND curptr < ;
            count _ -count;
            END
         ELSE
            FIND curptr > ;
      % act according to command %
         CASE char OF
            = 'C, = 'c: % content %
               FOR count DOWN UNTIL <= 0 DO
                  FIND CH;
            = 'E, = 'e: % end of statement %
               FIND SE(curptr) CH;
            = 'F, = 'f: % front of statement %
               FIND SF(curptr);
            = 'I, = 'i: % invisible %
               BEGIN
                  FOR count DOWN UNTIL <= 0 DO
                     FIND $NP $PT;
                  IF scnbck THEN FIND $NP;
               END;
            = 'L, = 'l: % link scan %
               FOR count DOWN UNTIL <= 0 DO
                  FIND CH C"(/"<3;
            = 'N, = 'n: % number %
               FOR count DOWN UNTIL <= 0 DO
                  BEGIN
                     FIND $D $(NOT D) ^pscan;
                     % call number delimiter routine to find end
                     of number %
                        ndr( $pscan, $tp1, $tp2 );
                     % set pointer to end of current number %
                        IF scnbck
```

```
                        THEN FIND tp1
                        ELSE FIND tp2 $(NOT D);
                    END;
              = 'V, = 'v: % visible %
                 BEGIN
                    FOR count DOWN UNTIL <= 0 DO
                       FIND $PT $NP;
                    IF scnbck THEN FIND $PT;
                 END;
              = 'W, = 'w: % word %
                 BEGIN
                    FOR count DOWN UNTIL <= 0 DO
                       FIND $LD $NLD;
                    IF scnbck THEN FIND $LD;
                 END;
              = LD:
                 RETURN(FALSE);
              ENDCASE RETURN(result);
        % update the parameters %
           FIND ^curptr tp > CH ^tp;
           result _ TRUE;
     END; % of main loop %
     % Return %
        RETURN;
     END.

(consearch)   % L: do content search for address evaluation  %
PROCEDURE (cont REF, curptr REF, srchtyp, count, domain % => T/F
%);                                                              1C6
     % Procedure description
        FUNCTION
           do content search for address evaluation
        ARGUMENTS
           cont    - STRING - content to search for
           curptr  - TEXT POINTER - where to start the search
           srchtyp - INTEGER - type of search (content, word)
           count   - INTEGER - how many to find
           dmain   - INTEGER - domain of search
        RESULTS
           TRUE if content found FALSE otherwise
        NON-STANDARD CONTROL
           none
        GLOBALS
           conreg
        %
     % Declarations %
        LOCAL tmp, onechar;
     % initialize %
        *conreg* _ *cont*;
        onechar _ (conreg.L = 1);
     % act according to the domain %
        IF domain = -1 THEN %no sdomain%
           FOR tmp _ 1 UP UNTIL > count DO
              BEGIN
                 IF onechar AND tmp>1 THEN BUMP curptr[1];
                 lookup( &curptr, $conreg, srchtyp );
```

```
            END
         ELSE % sdomain defined %
            FOR tmp _ 1 UP UNTIL > count DO
               BEGIN
                  IF onechar AND tmp>1 THEN BUMP curptr[1];
                  IF lookup( &curptr, $conreg, srchtyp) = endfile
                  THEN
                     BEGIN
                        IF (domain _ domain - 1) > 0 THEN
                           BEGIN
                              curptr _ getnxt(curptr);
                              BUMP DOWN tmp;
                           END
                        ELSE EXIT LOOP;
                     END;
               END;
      % Return %
         RETURN(curptr # endfile);
      END.

%************************************************
         Address Expression support routines
*************************************************%
   (adxerr)   % L: address expression error routine %
   PROCEDURE (errmsg REF % => Never %);                              1D1
      % Procedure description
         FUNCTION
            address expression error routine
         ARGUMENTS
            errmsg = STRING - error message
         RESULTS
            never returns
         NON-STANDARD CONTROL
            ABORTS with type gaderr
         GLOBALS
            none
         %
      % Declarations %
         LOCAL STRING locstr[199];
      % compose the message %
         *locstr* _ *errmsg*, " ?";
      % create a signal %
         ABORT( gaderr, $locstr );
      % Return %
         RETURN;
      END.

   (conspt)  % used for content and word searches %                  1D2
      % determines type of search, number of elements to be found,
      and scope of the search.%
      PROCEDURE(z1);
      LOCAL
         count, type, scope, tmp;
      REF z1;
      count _ type _ scope _ tmp _ 0;
      IF NOT FIND $(SP/TAB) '= THEN
```

```
      RETURN( contnt, 1, -1);
   FIND ^z1;
   CASE READC OF
      IN ['0, '9]:
         BEGIN
         tmp _ gadnum(&z1);
         REPEAT CASE;
         END;
      = 'C, ='c:
         IF NOT type THEN
            BEGIN
            FIND ^z1;
            IF tmp THEN count _ tmp := 0 ELSE count _ 1;
            type _ IF scope THEN cntls ELSE contnt;
            IF scope THEN RETURN( type, count, scope )
            ELSE REPEAT CASE;
            END
         ELSE
            BEGIN
            FIND z1;
            RETURN( type, count, IF scope THEN scope ELSE -1 );
            END;
      = 'W, ='w:
         IF NOT type THEN
            BEGIN
            FIND ^z1;
            IF tmp THEN count _ tmp := 0 ELSE count _ 1;
            type _ IF scope THEN wordls ELSE wordtyp;
            IF scope THEN RETURN( type, count, scope )
            ELSE REPEAT CASE;
            END
         ELSE
            BEGIN
            FIND z1;
            RETURN( type, count, IF scope THEN scope ELSE -1 );
            END;
      = 'S, ='s:
         IF NOT scope THEN
            BEGIN
            FIND ^z1;
            IF tmp THEN scope _ tmp := 0 ELSE scope _ 1;
            CASE type OF
               = FALSE:  REPEAT CASE 2;
               = wordtyp:
                  RETURN( wordls, count, scope );
               = contnt:
                  RETURN( cntls, count, scope );
               ENDCASE err($"NLS System Error: CONSPT");
            END
         ELSE
            BEGIN
            FIND z1;
            RETURN(
               IF type THEN type ELSE contnt,
                  IF count THEN count ELSE 1,
                     scope );
```

```
            END;
      ENDCASE
         BEGIN
         FIND z1;
         RETURN(
            IF type THEN type ELSE contnt,
               IF count THEN count ELSE 1,
                  IF scope THEN scope ELSE -1 );
         END;
   END.

   %  %
```

```
(fstid)   % L: ;  get t-ptr to new file %
PROCEDURE (flnam REF % => 2 text-pointer words %);                    1D3
   % Procedure description
      FUNCTION
         get t-ptr to new file. handle journal files differently.
         used by caddexp
      ARGUMENTS
         flnam - STRING - filname to load. (in OS style)
      RESULTS
         text ppointer to origin of file
      NON-STANDARD CONTROL
         err called if something goes wrong
      GLOBALS
         none
      %
   % Declarations %
      LOCAL jnlfg;
      LOCAL TEXT POINTER tptr, z1, z2;
      LOCAL STRING name[49];
   %Check to see if file is a journal file; if so, save off
   number%
      IF FIND SF(*flnam*) ('<[">]/) ^z1 4$D ^z2 THEN
         BEGIN
         *name* _ 'J, z1 z2;
         jnlfg _ TRUE;
         END
      ELSE jnlfg _ FALSE;
   % load the file %
      tptr _ origin;
      tptr.stfile _ cloafil(&flnam);
         %cloafil changes flnam%
   %Check to see if file is a journal message file%
      IF jnlfg AND FIND SF(*flnam*) ('<[">]/) "JRNL" THEN % find
      the name %
         lookup($tptr, $name, nametyp);
   % did we make it %
      IF tptr = endfile THEN
         BEGIN
            *name* _ z1 z2;
            adxerr($name);
         END;
   % Return %
      RETURN (tptr, 1);
   END.

(gadnum) PROCEDURE( tp REF);                                          1D4
   % extract and evaluate number pointed to by tp  -- return 1 if
   no number found  %
   LOCAL TEXT POINTER tp1;
   LOCAL STRING locstr[49];
   FIND tp < $D ^tp1 > $D ^tp;
   *locstr* _ tp1 tp;
   RETURN( IF locstr.L = 0 THEN 1 ELSE VALUE($locstr) );
   END.

(gdlit2) PROCEDURE(ast REF, stopchar);                                1D5
```

```
    % append characters to ast from READC until encounter a
    stopchar or the end of input.%
    LOCAL STRING char[1];
    LOCAL TEXT POINTER tp1, tp2;
    *char* _ stopchar;
    FIND ^tp1 [*char* ^tp2 _tp2/ENDCHR ^tp2];
    *ast* _ tp1 tp2;
    RETURN;
    END.

(getpr)   % call procedure count times %
PROCEDURE(type, count, curptr REF);                                1D6
   % Procedure description
      FUNCTION
         call a procedure count times while updating curptr. used
         by caddexp
      ARGUMENTS
         type - INTEGER - which procedure to call
         count - INTEGER = how many times
         curptr - TEXT POINTER - to be updated
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL stid, proc REF;
   % need to do at all? %
      IF count = 0 THEN RETURN;
   % need to do inverse? %
      IF count < 0 THEN %do the inverse%
         BEGIN
         type _ CASE type OF
            = cwsuccessor : cwpredecessor;
            = cwpredecessor : cwsuccessor;
            = cwdown :   cwup;
            = cwup   : cwdown;
            = cwhead : cwtail;
            = cwtail : cwhead;
            = cwnext : cwback;
            = cwback : cwnext;
            ENDCASE type;
         count _ -count;
         END;
   % set up procedure call %
      &proc _ CASE type OF
         = cwpredecessor : $getprd;
         = cwdown : $getsub;
         = cwup   :  $getup;
         = cwnext : $getnxt;
         = cwback : $getbck;
         ENDCASE type;
   % call the appropriate procedure %
      curptr[1] _ 1;
```

```
      CASE type OF
         = cwhead: curptr _ gethed(curptr);
         = cwtail: curptr _ getail(curptr);
         = cwend:  curptr _ getend(curptr);
         = cwsuccessor:
            BEGIN
               stid _ getail(curptr);
               FOR count DOWN UNTIL <= 0 DO
                  IF curptr = stid THEN EXIT LOOP
                  ELSE curptr _ getsuc(curptr);
            END;
         ENDCASE
            FOR count DOWN UNTIL  <= 0 DO
               IF (stid _ proc(curptr)) = endfile THEN EXIT LOOP
               ELSE curptr _ stid;
   RETURN;
   END.

(lkmkr) PROCEDURE (astrng, fileno); %translate marker name to
t-ptr%                                                          1D7
   %lkmkr converts a character pointer name (up to five
   characters) from the passed astring to a T-pointer for that
   marker.  It returns (ENDFIL) if that pointer does not exist.%
   %----------------------%
   LOCAL tmpnam, count, char, end, marker, flhd, stid;
   REF astrng, marker;
   stid _ origin;
   flhd _ filehead[stid.stfile _ fileno] - $filhed;
   &marker _ flhd + $mkrtb;
   end _ &marker + [flhd + $mkrtbl]*mkrl;
   IF [flhd + $mkrtbl] <= 0 OR astrng.L <= empty THEN
      RETURN(endfil);
   tmpnam _ 0;
   count _ 1;
   CCPOS SF(*astrng*);
   LOOP %extract marker tmpnam from a-string%
      BEGIN
      IF (char _ READC) = ENDCHR THEN EXIT;
      CASE count OF
         = 1: tmpnam.chr0 _ char;
         = 2: tmpnam.chr1 _ char;
         = 3: tmpnam.chr2 _ char;
         = 4: tmpnam.chr3 _ char;
         = 5: tmpnam.chr4 _ char;
         ENDCASE EXIT;
      BUMP count;
      END;
   DO IF marker.mkname = tmpnam THEN
      BEGIN   %depends on the extra bit being cleared by any
      routine adding markers to the marker table%
      stid.stpsid _ marker.mkpsid;
      RETURN (stid, marker.mkccnt)
      END
   UNTIL (&marker _ &marker + mkrl) = end;
   RETURN(endfil); %no such marker%
   END.
```

```
%*************************************************
                statement name lookup
*************************************************%
  (namelook)PROC(stid, astr);                                    1E1
    %This routine accepts an stid and astring, an finds the named
    statemet in the file indicated by the stid, retruning the stid
    oof the statemet or endfil.  Uses non-sequential lookup.
    Copies name astring into local string so a literal may be used
    in the call%
    LOCAL TEXT POINTER z1;
    LOCAL STRING namest[40];
    REF astr;
    FIND SF(stid) ^z1;
    *namest* _ *astr*;
    lookup($z1, $namest, nametyp);
    RETURN(z1);
    END.

  (lookup)PROCEDURE(ptr, astrng, type);                          1E2
    %THis routine accepts a pointer, string, and type, and does a
    search through the file indicated by the pointer for the
    statemnt indicated by the string and type as follows:
      type = nametyp
        searches the file for the first statement found with the
        indicated name.
        Does a non-sequential search.
        The string is modified.
        Returns the stid of the statement as a result and in the
        pointer, or endfil on failure.
      type = nxtname
        Same as name, but starts from the place in the ring
        indicated by the stid in ptr.
        Note that this also is a non-sequential search
      type = seqname:
        Does a sequential search of the file for the next
        statemet (beginning with the one following the one
        indicated by ptr) with the indicated name.
        Returns stid in ptr, or endfil in ptr.
      type = braname
        Same as seqname, but search is restricted to branch
        headed by ptr.
      type = contnt
        Does a sequential search of the file fo a statement with
        the content in string.
        Search starts with the character following the one
        indicated by ptr
        Returns same as seqname.
      type = contls
        Same as content, but looks only in the current statement
      type = wordtyp
        Searches for word in string in a manner equivalent to
        content.
        Returns same as content.
      type = wordls
        Same as wordtyp, but looks only in the current statement
```

```
      type = sid
         &astrng is assumed to be an sid, and te file is searched
         for a statement with a matching sid.
         Returns same as name.
   %
   %--------------------%
   LOCAL nhash, count, sidval, p1scn, p2scn, stid;
   REF astrng, ptr;
   IF ptr = endfil THEN RETURN(endfil);
   CASE type OF
      = nametyp, =nxtname, =extname:
         BEGIN
         astruc(&astrng);
         nhash _ hash(&astrng);
         stid _ IF type = nxtname THEN ptr ELSE 0;
         WHILE (stid _ lkupfast(ptr, stid, nhash, rnameh)) #
         endfil DO
            BEGIN
            CCPOS SF(stid);
            xtrnam($lkupreg, $swork, -1, 0);
            IF *lkupreg* = *astrng* THEN
               RETURN(ptr _ stid);
            END;
         RETURN(ptr _ endfil);
         END;
      = seqname:
         BEGIN
         astruc(&astrng);
         nhash _ hash(&astrng);
         ptr[1] _ 1;
         ptr _ getnxt(ptr);
         END;
      = braname:
         BEGIN
         ptr[1] _ 1;
         RETURN(ptr _ namingrp(ptr, ptr, &astrng, 1000));
         END;
      = sid:
         RETURN(ptr _ lkupfast(ptr, 0, &astrng, rsid));
      ENDCASE IF ptr[1]=empty THEN ptr[1]_1;
   UNTIL ptr = endfil DO
      BEGIN
      IF inptrf THEN %have a rubout%
         BEGIN
         ptr _ endfil;
         RETURN;
         END;
      CASE type OF
         =seqname:
            IF getnam(ptr) = nhash THEN
               BEGIN
               CCPOS SF(ptr);
               <UTILTY, xtrnam>($lkupreg, $swork, -1, 0);
               IF <UTILTY, compas>($lkupreg, &astrng) THEN
               RETURN;
               END;
```

```
            =contnt:
               IF FIND ptr > [*astrng*] ^ptr _ptr THEN RETURN;
            =contls: %search is limited to a single statement%
               BEGIN
               IF NOT FIND ptr > [*astrng*] ^ptr _ptr THEN ptr _
               endfil;
               RETURN;
               END;
            =wordtyp:
               BEGIN
               CCPOS ptr;
               LOOP
                  IF FIND [*astrng*] ^ptr _ptr < THEN
                     BEGIN
                     count _ astrng.L;
                     UNTIL (count _ count-1) < empty DO FIND CH;
                     IF FIND ^plscn -LD AND ptr > CH -LD THEN
                     RETURN;
                     IF NOT FIND > plscn CH THEN EXIT;
                        % this can EXIT if astrng is empty and
                        have reached end of statement %
                     END
                  ELSE EXIT;
               END;
            =wordls:
               BEGIN
               CCPOS ptr;
               LOOP
                  IF FIND [*astrng*] ^ptr _ptr < THEN
                     BEGIN
                     count _ astrng.L;
                     UNTIL (count _ count-1) < empty DO FIND CH;
                     IF FIND ^plscn -LD AND ptr > CH -LD THEN
                     RETURN;
                     IF NOT FIND > plscn CH THEN
                        % this can RETURN if astrng is empty and
                        have reached end of statement %
                        BEGIN
                        ptr _ endfil;
                        RETURN;
                        END;
                     END
                  ELSE
                     BEGIN
                     ptr _ endfil;
                     RETURN;
                     END;
               END;
            ENDCASE err($"NLS system error");
         ptr _ getnxt(ptr);
         ptr[1] _ 1;
         END;
      RETURN
      END.

(lkupfast)PROC(stid, start, value, field);                    1E3
```

```
    LOCAL rnb, rnptr, rnt, pgindx, rngbkend, rngbkptr, fileno;
    REF rnptr, rngbkptr;
    fileno _ stid.stfile;
    rnt _ (rnb _ filehead[fileno]+$rngst-$filhed) + rngm;
    &rnptr _ rnb+start.stblk-1;
    IF start THEN
       BEGIN
       IF (start _ start.stwc + ringl) >= blksiz THEN  %Gone over
       to te next block%
          BEGIN
          BUMP &rnptr;
          start _ 0;
          END
       ELSE start _ start - fbhdl;
       END;
    WHILE (&rnptr _ &rnptr+1) < rnt DO
       IF rnptr.rfexis THEN
          BEGIN
          IF (pgindx _ rnptr.rfcore) = 0 THEN
             pgindx _ lodrfb(&rnptr-rnb+rngbas, rngtyp, fileno);
          rngbkend _ (&rngbkptr _ crpgad[pgindx] + fbhdl) +
          blksiz;
          IF start THEN &rngbkptr _ &rngbkptr + start := 0; %add
          in displacement if it's there%
          WHILE &rngbkptr < rngbkend DO
             IF rngbkptr.field = value THEN
                BEGIN
                stid.stblk _ &rnptr-rnb;
                stid.stwc _ &rngbkptr-crpgad[pgindx];
                RETURN(stid);
                END
             ELSE &rngbkptr _ &rngbkptr + ringl;
          END;
    RETURN(endfil);
    END.

(namingrp) %lookup name in file starting at stid and terminating
when through with the branch headed by lbstid, looking down
levels levels.%                                                  1E4
    % This procedure finds the statement whose name is in namstr.
    the search is restricted to that part of the file begining
    with STID through the branch headed by LBSTID, and within
    LEVELS levels below STID. %
    %--------------------%
    PROC(stid, lbstid, namstr, levels);
    LOCAL rng, nhash, save;
    LOCAL STRING locstr[100];
    REF namstr, rng;
    IF stid = endfil THEN RETURN(endfil);
    astruc(&namstr);
    nhash _ hash(&namstr);
    save _ rubabt := FALSE;
    LOOP
       BEGIN
       IF getnmf(stid) AND getnam(stid) = nhash THEN
          BEGIN
```

```
            CCPOS SF(stid);
            xtrnam($locstr, $swork, -1, 0);
            IF *namstr* = *locstr* THEN
               BEGIN
               rubabt _ save;
               RETURN(stid);
               END;
            END;
         IF levels > 0 AND (stid := getsub(stid)) # stid THEN
            BUMP DOWN levels
         ELSE
            LOOP
               IF getftl(stid) THEN
                  BEGIN
                  BUMP levels;
                  IF stid = lbstid OR inptrf THEN EXIT LOOP 2;
                  stid _ getsuc(stid); %GETSUC RETURNS THE UP IF
                  TAIL%
                  END
               ELSE
                  BEGIN
                  IF stid = lbstid OR inptrf OR stid.stpsid = origin
                  THEN
                     EXIT LOOP 2;
                  stid _ getsuc(stid);
                  EXIT LOOP;
                  END;
            END;
      rubabt _ save;
      RETURN(endfil);
      END.

%*** leftovers from old caddexp ***%
   % the following procedures were used by other then caddexp in
   NLS8.5;  CADDEXPP doesn't use them any more but theyy are used
   elsewhere in NLSBE.  Should be removed from the BE as soon as
   possible %
   (gadname) PROCEDURE(ptr, ast, type);                              1F2
      LOCAL TEXT POINTER tp1, tp2, tp3;
      REF ptr, ast;
      FIND ^tp1;
      nmdr($tp1, $tp2, $tp3);
      *ast* _ tp2 tp3;
      specreg(&ast, type, &ptr);
      FIND tp3 >;
      RETURN;
      END.

      %  %
```

```
(ntstid)  %get t-ptr to new file and address expression%        1F3
   PROCEDURE(fnmstng, stnstg, vspstg, da);
   %Given two strings, the first containing a file name, the
   second an address experssion, this routine will open the file,
   and return the corresponding stid and character count. Branch
   only viewspec is added to the viewspec string if this is a
   journal message.  DA is needed for address expression
   evaluation.%
   %-------------%
   LOCAL vs1, vs2, cacode, useqgen, fno, jnlfg;
   LOCAL TEXT POINTER tptr, z1, z2;
   LOCAL STRING lkmkst[15], lkfnst[50];
   REF fnmstng, stnstg, vspstg, da;
   %Check to see if file is a journal file; if so, save off
   number%
      IF FIND SF(*fnmstng*) ("<[">]/) ^z1 4$D ^z2 THEN
         BEGIN
         *lkfnst* _ "J, z1 z2;
         jnlfg _ TRUE;
         END
      ELSE jnlfg _ FALSE;
   tptr _ origin;
   tptr.stfile _ fno _ cloafil(&fnmstng);
      %cloafil changes fnmstng%
   tptr[1] _ 1;
   %Check to see if file is a journal message file%
      IF jnlfg AND FIND SF(*fnmstng*) ("<[">]/) "JRNL" THEN
         BEGIN
         *stnstg* _ *lkfnst*;
         END
      ELSE jnlfg _ FALSE;
   FIND SF(*stnstg*) ^z1 SE(*stnstg*) ^z2;
   vs1 _ caddexp($z1, $z2, &da, $tptr : vs2, cacode, useqgen);
   da.davspec _ vs1;
   da.davspc2 _ vs2;
   da.dacacode _ cacode;
   da.dausgcod _ useqgen;
   RETURN (tptr, tptr[1]);
   END.

   %  %
```

```
(specreg)                                                        1F4
   %Spec a register.  This procedure converts a string or a
   statement identifier to a t-pointer.  The conversion algorithm
   depends on the first character in the register, and on the
   parameter passed as the second argument.  If a string is being
   matched, the routine expects as a third argument the stid at
   which the search should begin.
      If the first character is a number, and a name is being
      speced, the register is assumed to contain a statement
      number, and FECHUX is used to convert the A-string to a
      STID.  Otherwise the register is assumed to contain either
      a word or a string to be matched.
      If TYPE (the second argument) is
      1, the register is assumed to contain a name;
      2, the register is assumed to contain a word;
      3 or 6, the register is assumed to contain a string;
      4, the register is assumed to contain a statement
      identifier;
      In all of these cases LOOKUP is called to find the STID.%
   %---------------%
   PROCEDURE(astrng, type, ptr);
   REF astrng, ptr;
   IF astrng.L = empty
      THEN
         BEGIN
         ptr.stpsid _ origin;
         ptr[1] _ 1;
         RETURN;
         END;
   IF type = nametyp OR type = segname OR type = nxtname OR type
   = extname
      THEN
         BEGIN %number, sid or name%
         ptr[1] _ 1;
         IF *astrng*[ 1 ] IN ['0, '9]
            THEN
               BEGIN %number or SID%
               IF *astrng*[ 1 ] = '0 AND astrng.L > 1  % sid
               given %
                  THEN
                     BEGIN
                     *astrng* _ *astrng*[2 TO astrng.L]; %delete
                     '0 %
                     lookup(&ptr, cvsno(&astrng), sid); %lookup
                     sid%
                     RETURN;
                     END;
               ptr _ fechux(&astrng, ptr.stfile);
               RETURN;
               END;
         END;
   lookup(&ptr, &astrng, type);
   RETURN;
   END.

   %  %
```

```
   (cvsno) PROCEDURE(astr); %convert a-string to number%          1F5
      %An A-string containing digits can be converted to a binary
      integer by calling CVSNO with the A-string address.
      Theconversion is done to the base 10, and all characters are
      assumed to be digits, and are not checked.  The A-string is
      not changed, and the integer is returned.  If the first
      character is a  minus sign, the number will be converted to a
      negative number correctly.%
      %-----------------%
      LOCAL
         chrcnt, %character count for char readout%
         negnum, %negative number flag%
         number; %cell in which the number is constructed%
      REF astr;
      number _ negnum _ 0;
      chrcnt _ empty + 1;
      IF *astr*[chrcnt] = '- THEN BUMP chrcnt, negnum;
      DO number _ number*10 + (*astr*[chrcnt] - '0)
      UNTIL (chrcnt := chrcnt + 1) = astr.L;
      RETURN(IF negnum THEN -number ELSE number);
      END.

FINISH
(old) stuff. put here by ROM 2-Jun-78 09:57                          3
   %Address Expression evaluation%
      (caddexp) %***%   %Core NLS Address Expression evaluation
      Command%                                                     3A1
         %given two text pointers to a text string, a pointer to a
         display area record (for viewspecs and return rings), and a
         text pointer into a file, this routine will evaluate the
         expression relative to the text pointer and update the text
         pointer.  It will also return updated viewspec words,
         content-analyzer-program, and user-seqgen program addresses.
         It will also return 0 or the address of the relevant statement
         return ring if the last thing done is a .fr  %

         PROCEDURE(t1, t2, da, ptr);
         LOCAL
            savslh,      % save value of slashflg %
            rhstn,       % host number from links %
            fnlsls,      % for doing final / %
            sdomain,     % domain for content searches %
            savs1,savs2,         % saved viewspecs for errors %
            cacode,      % saved address of content analyzer program %
            useggen,     % saved address of sequence generator program %
            proc,
            dir,         % direction for scan relations %
            char,        % for main loop dae parsing %
            count,       % for scan and positional relationships %
            scnbck,
            sring,       % adr stmnt return ring for file return, etc. %
            fname,       % adr file return fing tor file return %
            tmp,
            f1, % file number for jump name external file %
            enftype,  %TRUE, look for external name link in file
            origin; FALSE, go to user-options only%
```

```
    ldstr[35];  % data structure for link parsing %
LOCAL STRING
    jnestr[200],        % for jump name external file %
    locstr[200],        % to hold dae locally %
    st1[100], st2[100], st3[100], st4[300]; %for link parsing%
LOCAL TEXT POINTER
    pscan, tp1, tp2, tp9, z1;
REF ptr, da, t1, t2, f1;

% initialize %
    savslh _ slashflg := FALSE;
    fnlsls _ fakesrr _ FALSE;
%save value of viewspec data%
    savs1 _ da.davspec;
    savs2 _ da.davspc2;
    cacode _ da.dacacode;
    useggen _ da.dausgcod;
% cleanup on errors %
    INVOKE (sigclean);
% copy dae to minimize page faulting, etc. %
    *locstr* _ t1 t2;
FIND SF(*locstr*) ^z1 ^tp9 >;
LOOP
    BEGIN
    IF irptrf THEN GOTO caeerror;
    count _ 1;
    FIND z1 >;
    *errwrk* _ char _ READC;
    FIND ^z1;
    CASE char OF
       = SP, = TAB, = EOL, = CR, = LF: REPEAT LOOP;
       = ENDCHR: NULL;
       ENDCASE fnlsls _ fakesrr _ FALSE;
    CASE char OF
       = SP, = TAB, = EOL, = CR, = LF:  NULL;
       ="(:   % positional relation %
          LOOP
             BEGIN
             FIND z1 >;
             *errwrk* _ char _ READC;
             FIND ^z1;
             count _ 1;
             CASE char OF
                IN ["0, "9]:
                   BEGIN
                   FIND < CH >;
                   count _ gadnum();
                   *errwrk* _ char _ READC;
                   FIND ^z1;
                   CASE char OF
                      #L: GOTO caeerror;
                      ENDCASE REPEAT CASE 2(char);
                   END;
                ="B, ="b:  % back %  %SHOULD USE SEQGEN%
                   BEGIN
                   fakesrr _ FALSE;
```

```
            getpr($getbck, count, &ptr);
            END;
        ='C, ='c:  % next occurrence of content %
            BEGIN
            fakesrr _ FALSE;
            *errwrk* _ '", *conreg*, '";
            tmp _ 1;
            srchtype _ contnt;
            FOR tmp UP UNTIL > count DO
               BEGIN
               IF (tmp > 1) AND (conreg.L = 1) THEN BUMP
               ptr[1];
               specreg( $conreg, contnt, &ptr);
               END;
            END;
        ='D, ='d:  % down %
            BEGIN
            fakesrr _ FALSE;
            getpr($getsub, count, &ptr);
            END;
        ='E, ='e:  % end %
            BEGIN
            fakesrr _ FALSE;
            ptr _ getvnd(ptr, 1000 %large # of levels%);
            ptr[1] _ 1;
            END;
        ='F, ='f:  % file return %
            BEGIN
            *errwrk* _ char _ READC;
            CASE char OF
               ='R, ='r:  NULL;
               ENDCASE GOTO caeerror;
            FIND ^z1;
            INVOKE (badfrr);
            fname _ readfrring(da.dalink, count :
            fakesrr);
            DROP (badfrr);
            FIND SF(*[fname]*) ^tp1;
            IF (rhstn _ lnbfls($tp1, 0, $jnestr)) #
            lhostn THEN
               err($"Remote File Manipulation Not
               Implemented Yet");
            ptr _ readsrring(fakesrr, 0 : ptr[1],
            da.davspec, da.davspc2);
            ptr.stfile _ cloafil($jnestr);
            END;
        ='H, ='h:  % head %
            BEGIN
            fakesrr _ FALSE;
            getpr($gethed, count, &ptr);
            END;
        ='L, ='l:  % indirect link %
            BEGIN
            fakesrr _ FALSE;
            ptr[1] _ MAX(fchtxt(ptr), ptr[1]);
            IF (rhstn _ lnkpspc(
```

```
            count, &ptr, $st1, $st2, $st3, $st4,
            $ldstr)) # lhostn THEN
               err($"Remote File Manipulation Not
               Implemented Yet");
         IF ldstr[fe+1] > ldstr[fs+1] THEN %filename%
            ptr _ nfstid ($st2, $st3, $st4, &da :
            ptr[1])
         ELSE %old file%
            IF st3.L THEN % address expression %
               BEGIN
               FIND SF(*st3*) ^tp1 SE(*st3*) ^tp2;
               da.davspec _ caddexp( $tp1, $tp2, &da,
               &ptr : da.davspc2, da.dacacode,
               da.dausqcod );
               END;
         IF st4.L # empty THEN feedlt(&da, $st4);
         END;
      ='N, ='n:  % next %  %SHOULD USE SEQGEN%
         BEGIN
         fakesrr _ FALSE;
         getpr($getnxt, count, &ptr);
         END;
      ='O, ='o:  % origin %
         BEGIN
         fakesrr _ FALSE;
         ptr.stpsid _ origin;
         ptr[1] _ 1;
         END;
      ='P, ='p:  % predecessor %
         BEGIN
         fakesrr _ FALSE;
         getpr($getprd, count, &ptr);
         END;
      ='R, ='r:  % return %
         BEGIN
         fakesrr _ FALSE;
         INVOKE (badfrr);
         readfrring(da.dalink, 0 : sring);
         DROP (badfrr);
         ptr _ readsrring(sring, count : ptr[1],
         da.davspec, da.davspc2);
         END;
      ='S, ='s:  % succ %
         BEGIN
         fakesrr _ FALSE;
         getpr($getsuc, count, &ptr);
         END;
      ='T, ='t:  % tail %
         BEGIN
         fakesrr _ FALSE;
         getpr($getail, count, &ptr);
         END;
      ='U, ='u:  % up %
         BEGIN
         fakesrr _ FALSE;
         getpr($getup, count, &ptr);
```

```
                    END;
                ="W, ="w:  % next occurrence of word %
                    BEGIN
                    fakesrr _ FALSE;
                    *errwrk* _ "", *conreg*, "", "=w";
                    tmp _ 1;
                    srchtype _ wordtyp;
                    FOR tmp UP UNTIL > count DO
                        BEGIN
                        IF (tmp > 1) AND (conreg.L = 1) THEN BUMP
                        ptr[1];
                        specreg( $conreg, wordtyp, &ptr);
                        END;
                    END;
                # LD:   % end of structural relation %
                    BEGIN
                    CASE char OF
                        = ENDCHR:  NULL;
                        ENDCASE FIND z1 < CH ^z1;
                    REPEAT LOOP 2;
                    END;
                ENDCASE GOTO caeerror;
            END;
        = "-:     % scan relation or link %
            CASE READC OF
                = "-:       % link %
                    BEGIN
                    FIND < CH >;
                    GOTO cadlnk;
                    END;
                = ENDCHR:  GOTO caeerror;
                ENDCASE     % scan relation %
                    BEGIN
                    FIND < CH >;
                    GOTO cadscr;
                    END;
        ="+:    % scan relation %
            (cadscr):                                          3A1N8D1
            BEGIN
            dir _ IF char = "+ THEN 1 ELSE -1;
            LOOP
                BEGIN
                FIND z1 >;
                *errwrk* _ char _ READC;
                FIND ^z1;
                count _ dir;
                CASE char OF
                    IN ["0, "9]:
                        BEGIN
                        FIND < CH >;
                        count _ gadnum() * dir;
                        *errwrk* _ char _ READC;
                        FIND ^z1;
                        CASE char OF
                            #L: GOTO caeerror;
                            ENDCASE REPEAT CASE 2(char);
```

```
    END;
='C, ='c: %character scan%
    BEGIN
    count _ gaddir(&ptr, count);
    UNTIL (count _ count-1) < 0 DO FIND CH;
    FIND ^ptr;
    END;
='E, ='e: %statement end%
    FIND SE(ptr) < CH ^ptr;
='F, ='f: %statement front%
    FIND SF(ptr) ^ptr;
='I, ='i: %invisible scan%
    BEGIN
    count _ gaddir(&ptr, count : scnbck);
    UNTIL (count _ count-1) < 0 DO
       FIND $NP $PT;
    IF scnbck THEN FIND $NP;
    FIND ^ptr;
    END;
='L, ='l: %link scan%
    BEGIN
    count _ gaddir(&ptr, count : scnbck);
    UNTIL (count _ count-1) < 0 DO
       FIND CH '( / '<];
    FIND ^ptr;
    END;
='N, ='n: %number scan%
    BEGIN
    count _ gaddir(&ptr, count : scnbck);
    UNTIL (count _ count-1) < 0 DO
       BEGIN
       FIND $D $(NOT D) ^pscan;
       % call number delimiter routine to find
       end of number %
          ndr( $pscan, $tp1, $tp2 );
       % set pointer to end of current number %
          IF scnbck
             THEN FIND tp1
             ELSE FIND tp2 $(NOT D);
       END;
    FIND ^ptr;
    END;
='V, ='v: %visible scan%
    BEGIN
    count _ gaddir(&ptr, count : scnbck);
    UNTIL (count _ count-1) < 0 DO
       FIND $PT $NP;
    IF scnbck THEN FIND $PT;
    FIND ^ptr;
    END;
='W, ='w: %word scan%
    BEGIN
    count _ gaddir(&ptr, count : scnbck);
    UNTIL (count _ count-1) < 0 DO
       FIND $LD $NLD;
    IF scnbck THEN FIND $LD;
```

```
                    FIND ^ptr;
                    END;
                 # LD:   % end of scan relation %
                    REPEAT CASE 2(char);
              ENDCASE GOTO caeerror;
              END;
        END;
      ="/:   % slash %
        BEGIN
        ccurcon( &ptr, $errwrk );
        dismes (1, $errwrk);
        fnlsls _ TRUE;
        END;
      ="\:   % backslash %
        cprint (&da, ptr, ptr, stmtv);
      = "(, = "<:      %link %
        (cadlnk):                                          3A1N8G1
        BEGIN
        FIND < CH ^z1 ^tp1;
        lnkprs( $z1, $ldstr);
        ldstr[lfn] _ t1.stfile; %because copied bugged link%
        IF (rhstn _ lnkpspc( 1, 0, $st1, $st2, $st3, $st4,
        $ldstr)) # lhostn THEN
           err($"Remote File Manipulations Not Implemented
           Yet");
        %because copied bugged link%
           IF (st3.L = empty) AND
              t1.stfile AND (ldstr[fs+1] >= ldstr[fe+1]) THEN
                 *st3* _ "0, STRING(getsid(t1)), "+,
                 STRING(t1[1]), "c;
        z1 _ ldstr[le];  z1[1] _ ldstr[le+1];
        *errwrk* _ tp1 z1;
        IF ldstr[fe+1] > ldstr[fs+1] THEN  % file name %
           ptr _ nfstid( $st2, $st3, $st4, &da, : ptr[1])
        ELSE
           IF st3.L THEN % address expression %
              BEGIN
              FIND SF(*st3*) ^tp1 SE(*st3*) ^tp2;
              da.davspec _ caddexp( $tp1, $tp2, &da, &ptr :
              da.davspc2, da.dacacode, da.dausqcod );
              END;
        IF st4.L THEN feedlt( &da, $st4);
        END;
      ="": %content search%
        BEGIN
        FIND < CH ^tp1 > CH;
        *st1* _ NULL;
        gdlit2( $st1, "");
        FIND ^z1;
        *errwrk* _ tp1 z1;
        (srchpar):                                         3A1N8H7
           *conreg* _ *st1*;
           srchtype _ conspt( $z1 : count, sdomain);
           FIND ^z1;
           *errwrk* _ tp1 z1;
           CASE sdomain OF
```

```
                = -1:
                   BEGIN
                   tmp _ 1;
                   FOR tmp UP UNTIL > count DO
                      BEGIN
                      IF (tmp > 1) AND (st1.L = 1) THEN BUMP
                      ptr[1];
                      specreg( $st1, srchtype, &ptr);
                      END;
                   END;
                ENDCASE
                   BEGIN
                   tmp _ 1;
                   FOR tmp UP UNTIL > count DO
                      LOOP
                         BEGIN
                         FIND ptr ^tp1;
                         IF (tmp > 1) AND (st1.L = 1) THEN BUMP
                         ptr[1];
                         specreg( $st1, srchtype, &ptr);
                         IF ptr # endfil THEN EXIT LOOP;
                         IF (sdomain _ sdomain-1) THEN
                            ptr _ getnxt(tp1)
                         ELSE EXIT LOOP 2;
                         END;
                   END;
             END;
          ='' %apostrophe% : %character search%
             BEGIN
             FIND < CH ^tp1 > CH;
             *st1* _ char _ READC;
             *errwrk* _ '', char;
             FIND ^z1;
             GOTO srchpar;
             END;
          ='#: % marker %
             BEGIN
             *st1* _ NULL;
             gdlit2($st1, SP);
             FIND ^z1;
             *errwrk* _ *errwrk*, *st1*;
             ptr _ lkmkr($st1, ptr.stfile : ptr[1]);
             FIND z1 >;
             END;
          = '$,    % external name %
          = '*,    % next name %
          = '!,    % name in current branch %
          = LD, = '', = '-, = '@:    % name %
             BEGIN
             tmp _ ptr.stfile;      % save file no. for jump name
             external %
             IF z1[1] > 2 THEN *locstr*[z1[1]-2] _ SP;
             gadname(&ptr, $st2, (CASE char OF
                = '$: extname;
                = '*: seqname;
                = '!: braname;
```

```
            ENDCASE nametyp));
         FIND ^z1;
         IF char = "$ AND ptr = endfil THEN
            BEGIN
            enftype _ TRUE;  %look first in file origin%
            INVOKE (sigenf, recallenf);
            DISABLE (sigenf);
            (recallenf):                                        3A1N8N6E
            IF getenf( tmp, enftype, $jnestr) THEN
               BEGIN
               IF *jnestr* = *ojnestr* THEN
                  ptr _ jfnefln
               ELSE
                  BEGIN
                  IF jfnefln THEN
                     BEGIN
                     &fl _flntadr(jfnefln.stfile);
                     fl.flnoclos _ FALSE;
                     END;
                  *ojnestr* _ *jnestr*;
                  FIND SF(*jnestr*) ^tp1 SE(*jnestr*) ^tp2;
                  caddexp($tp1, $tp2, &da, &ptr);
                  jfnefln _ ptr;
                  &fl _flntadr(jfnefln.stfile);
                  fl.flnoclos _ TRUE;
                  END;
               IF ptr.stpsid = origin THEN *st1* _ *st2*, "
               .]"
               ELSE *st1* _ "[, *st2*, " .]";
               FIND SF(*st1*) ^tp1 SE(*st1*) ^tp2;
               IF enftype THEN ENABLE (sigenf);
               caddexp($tp1, $tp2, &da, &ptr);
               IF enftype THEN DROP (sigenf);
               END;
            END;
         IF ptr = endfil THEN *errwrk* _ *st2*;
         END;
      = ENDCHR:
         BEGIN
         slashflg _ savslh;
         IF (slashflg) AND (NOT fnlsls) THEN
            BEGIN
            ccurcon( &ptr, $errwrk );
            dismes (1, $errwrk);
            END;
         DROP (sigclean);
         RETURN (da.davspec := savs1,
            da.davspc2 := savs2,
               da.dacacode := cacode,
                  da.dausqcod := useqgen);
         END;
      ENDCASE
         GOTO caeerror;
   IF ptr = endfil THEN GOTO caeerror;
   END;  % of loop %
(caeerror): %error return%                                      3A1C
```

```
      slashflg _ savslh;
      da.davspec _ savs1;
      da.davspc2 _ savs2;
      da.dacacode _ cacode;
      da.dausgcod _ useggen;
      *errwrk* _ *errwrk*, " ?";  %append "? to error message%
      DROP (sigclean);
      ABORT(gaderr, $errwrk);

   (sigclean) CATCHPHRASE;                                          3A1P
      BEGIN
      DISABLE (sigclean);
      CASE SIGNALTYPE OF
         =aborttype:
         BEGIN
            slashflg _ savslh;
            da.davspec _ savs1;
            da.davspc2 _ savs2;
            da.dacacode _ cacode;
            da.dausgcod _ useggen;
         END;
      ENDCASE;
      CONTINUE;
      END;

   (sigenf) CATCHPHRASE;                                            3A1Q
      BEGIN
      DISABLE (sigenf);
      CASE SIGNALTYPE OF
         =aborttype:
            BEGIN
            enftype _ FALSE;  %skip file origin; use user profile
            only%
            TERMINATE;
            END;
         ENDCASE;
      CONTINUE;
      END;

   (badfrr) CATCHPHRASE();                                          3A1R
      BEGIN
      DISABLE (badfrr);
      CASE SIGNALTYPE OF
         = notetype :  NULL;
         = helptype :  NULL;
         = aborttype :
            IF SIGNAL = frremptyentry THEN
               err($"No current entry in file return ring.");
         ENDCASE;
      CONTINUE;
      END;

   END.
   % %
```

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

   (lkmkr) PROCEDURE (astrng, fileno); %translate marker name to
   t-ptr%                                                          3B1
      %lkmkr converts a character pointer name (up to five
      characters) from the passed astring to a T-pointer for that
      marker.  It returns (ENDFIL) if that pointer does not exist.%
      %----------------------%
      LOCAL tmpnam, count, char, end, marker, flhd, stid;
      REF astrng, marker;
      stid _ origin;
      flhd _ filehead[stid.stfile _ fileno] - $filhed;
      &marker _ flhd + $mkrtb;
      end _ &marker + [flhd + $mkrtbl]*mkrl;
      IF [flhd + $mkrtbl] <= 0 OR astrng.L <= empty THEN
         RETURN(endfil);
      tmpnam _ 0;
      count _ 1;
      CCPOS SF(*astrng*);
      LOOP %extract marker tmpnam from a-string%
         BEGIN
         IF (char _ READC) = ENDCHR THEN EXIT;
         CASE count OF
            = 1: tmpnam.chr0 _ char;
            = 2: tmpnam.chr1 _ char;
            = 3: tmpnam.chr2 _ char;
            = 4: tmpnam.chr3 _ char;
            = 5: tmpnam.chr4 _ char;
            ENDCASE EXIT;
         BUMP count;
         END;
      DO IF marker.mkname = tmpnam THEN
         BEGIN   %depends on the extra bit being cleared by any
         routine adding markers to the marker table%
         stid.stpsid _ marker.mkpsid;
         RETURN (stid, marker.mkccnt)
         END
      UNTIL (&marker _ &marker + mkrl) = end;
      RETURN(endfil); %no such marker%
      END.

   (consPt)  % used for content and word searches %                 3B2
      % determines type of search, number of elements to be found,
      and scope of the search.%
      PROCEDURE(z1);
      LOCAL
         count, type, scope, tmp;
      REF z1;

      count _ type _ scope _ tmp _ 0;
      IF NOT FIND $(SP/TAB) '= THEN
         RETURN( contnt, 1, -1);
      FIND ^z1;
      CASE READC OF
         IN ['0, '9]:
            BEGIN
```

```
            FIND < CH >;
            tmp _ gadnum();
            FIND ^z1;
            REPEAT CASE;
            END;
         = 'C, ='c:
            IF NOT type THEN
               BEGIN
               FIND ^z1;
               IF tmp THEN count _ tmp := 0 ELSE count _ 1;
               type _ IF scope THEN contls ELSE contnt;
               IF scope THEN RETURN( type, count, scope )
               ELSE REPEAT CASE;
               END
            ELSE
               BEGIN
               FIND z1;
               RETURN( type, count, IF scope THEN scope ELSE -1 );
               END;
         = 'W, ='w:
            IF NOT type THEN
               BEGIN
               FIND ^z1;
               IF tmp THEN count _ tmp := 0 ELSE count _ 1;
               type _ IF scope THEN wordls ELSE wordtyp;
               IF scope THEN RETURN( type, count, scope )
               ELSE REPEAT CASE;
               END
            ELSE
               BEGIN
               FIND z1;
               RETURN( type, count, IF scope THEN scope ELSE -1 );
               END;
         = 'S, ='s:
            IF NOT scope THEN
               BEGIN
               FIND ^z1;
               IF tmp THEN scope _ tmp := 0 ELSE scope _ 1;
               CASE type OF
                  = FALSE:  REPEAT CASE 2;
                  = wordtyp:
                     RETURN( wordls, count, scope );
                  = contnt:
                     RETURN( contls, count, scope );
                  ENDCASE err($"NLS System Error: CONSPT");
               END
            ELSE
               BEGIN
               FIND z1;
               RETURN(
                  IF type THEN type ELSE contnt,
                     IF count THEN count ELSE 1,
                        scope );
               END;
         ENDCASE
            BEGIN
```

```
        FIND z1;
        RETURN(
           IF type THEN type ELSE contnt,
              IF count THEN count ELSE 1,
                 IF scope THEN scope ELSE -1 );
        END;
END.

%  %
```

```
(gadname) PROCEDURE(ptr, ast, type);                                    383
   LOCAL TEXT POINTER tp1, tp2, tp3;
   REF ptr, ast;
   FIND ^tp1;
   nmdr($tp1, $tp2, $tp3);
   *ast* _ tp2 tp3;
   specreg(&ast, type, &ptr);
   FIND tp3 >;
   RETURN;
   END.

   %  %
```

```
(gaddir) PROCEDURE(ptr,count);                                  3B4
   REF ptr;
   IF count < 0 THEN
      BEGIN
      FIND ptr < ;
      RETURN ( -count, TRUE );
      END;
   FIND ptr > ;
   RETURN ( count, FALSE );
   END.

   %  %
```

```
(getpr) PROCEDURE(proc, count, ptr);                                    385
   LOCAL lptr;
   REF proc, ptr;
   IF count < 0 THEN %do the inverse%
      BEGIN
      &proc _ CASE &proc OF
         = $getsuc : $getprd;
         = $getprd : $getsuc;
         = $getsub : $getup;
         = $getup  : $getsub;
         = $gethed : $getail;
         = $getail : $gethed;
         = $getnxt : $getbck;
         = $getbck : $getnxt;
         ENDCASE &proc;
      count _ -count;
      END;
   UNTIL (count _ count-1) < 0 DO
      BEGIN
      IF &proc = $getsuc AND getftl(ptr) THEN EXIT;
         % thus successor of statement with no successor is NOP %
      IF (lptr _ proc(ptr)) = endfil THEN EXIT;  %NOP%
      ptr _ lptr;
      ptr[1] _ 1;
      END;
   RETURN;
   END.

   %  %
```

```
(gdiit2) PROCEDURE(ast, stopchar);                                  386
   % append characters to ast from READC until encounter a
   stopchar or the end of input.%
   LOCAL char;
   REF ast;
   LOOP
      BEGIN
      CASE char _ READC OF
         =ENDCHR, =stopchar:
            RETURN;
         ENDCASE
            *ast* _ *ast*, char;
      END;
   END.

   % %
```

```
(gadnum) PROCEDURE;                                                    3B7
   % extract and evaluate number from current READC source.  %
   LOCAL count, char;
   IF (char _ READC) NOT IN [^0, ^9] THEN
      BEGIN
      IF char NOT= ENDCHR THEN FIND < CH >;
      RETURN(1);
      END;
   count _ char - ^0;
   WHILE (char _ READC) IN [^0, ^9] DO
      count _ count*10 + char - ^0;
   IF char NOT= ENDCHR THEN FIND < CH >;
   RETURN(count);
   END.

   % %
```

```
%execute link%
   (nfstid)  %get t-ptr to new file and address expression%   3B8A
      PROCEDURE(fnmstng, stnstg, vspstg, da);
      %Given two strings, the first containing a file name, the
      second an address experssion, this routine will open the
      file, and return the corresponding stid and character
      count. Branch only viewspec is added to the viewspec string
      if this is a journal message.  DA is needed for address
      expression evaluation.%
      %-------------%
      LOCAL vs1, vs2, cacode, useqgen, fno, jnlfg;
      LOCAL TEXT POINTER tptr, z1, z2;
      LOCAL STRING lkmkst[15], lkfnst[50];
      REF fnmstng, stnstg, vspstg, da;
      %Check to see if file is a journal file; if so, save off
      number%
         IF FIND SF(*fnmstng*) ('<[">]/) ^z1 4$D ^z2 THEN
            BEGIN
            *lkfnst* _ "J, z1 z2;
            jnlfg _ TRUE;
            END
         ELSE jnlfg _ FALSE;
      tptr _ origin;
      tptr.stfile _ fno _ cloafil(&fnmstng);
         %cloafil changes fnmstng%
      tptr[1] _ 1;
      %Check to see if file is a journal message file%
         IF jnlfg AND FIND SF(*fnmstng*) ('<[">]/) "JRNL" THEN
            BEGIN
            *stnstg* _ *lkfnst*;
            END
         ELSE jnlfg _ FALSE;
      FIND SF(*stnstg*) ^z1 SE(*stnstg*) ^z2;
      vs1 _ caddexp($z1, $z2, &da, $tptr : vs2, cacode, useqgen);
      da.davspec _ vs1;
      da.davspc2 _ vs2;
      da.dacacode _ cacode;
      da.dausqcod _ useqgen;
      RETURN (tptr, tptr[1]);
      END.

      %  %
```

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

   (namelook)PROC(stid, astr);                                        3B9A
      %This routine accepts an stid and astring, an finds the
      named statemet in the file indicated by the stid, retruning
      the stid oof the statemet or endfil.  Uses non-sequential
      lookup.  Copies name astring into local string so a literal
      may be used in the call%
      LOCAL TEXT POINTER z1;
      LOCAL STRING namest[40];
      REF astr;
      FIND SF(stid) ^z1;
      *namest* _ *astr*;
      lookup($z1, $namest, nametyp);
      RETURN(z1);
      END.

      %  %
```

```
(lookup)PROCEDURE(ptr, astrng, type);                              3B9B
   %THis routine accepts a pointer, string, and type, and does
   a search through the file indicated by the pointer for the
   statemnt indicated by the string and type as follows:
      type = nametyp
         searches the file for the first statement found with
         the indicated name.
         Does a non-sequential search.
         The string is modified.
         Returns the stid of the statement as a result and in
         the pointer, or endfil on failure.
      type = nxtname
         Same as name, but starts from the place in the ring
         indicated by the stid in ptr.
         Note that this also is a non-sequential search
      type = segname:
         Does a sequential search of the file for the next
         statemet (beginning with the one following the one
         indicated by ptr) with the indicated name.
         Returns stid in ptr, or endfil in ptr.
      type = braname
         Same as segname, but search is restricted to branch
         headed by ptr.
      type = contnt
         Does a sequential search of the file fo a statement
         with the content in string.
         Search starts with the character following the one
         indicated by ptr
         Returns same as segname.
      type = cont1s
         Same as content, but looks only in the current
         statement
      type = wordtyp
         Searches for word in string in a manner equivalent to
         content.
         Returns same as content.
      type = word1s
         Same as wordtyp, but looks only in the current
         statement
      type = sid
         &astrng is assumed to be an sid, and te file is
         searched for a statement with a matching sid.
         Returns same as name.
   %
   %-------------------%
   LOCAL nhash, count, sidval, p1scn, p2scn, stid;
   REF astrng, ptr;
   IF ptr = endfil THEN RETURN(endfil);
   CASE type OF
      = nametyp, =nxtname, =extname:
         BEGIN
         astruc(&astrng);
         nhash _ hash(&astrng);
         stid _ IF type = nxtname THEN ptr ELSE 0;
         WHILE (stid _ lkupfast(ptr, stid, nhash, rnameh)) #
         endfil DO
```

```
                BEGIN
                CCPOS SF(stid);
                xtrnam($lkupreg, $swork, -1, 0);
                IF *lkupreg* = *astrng* THEN
                   RETURN(ptr _ stid);
                END;
             RETURN(ptr _ endfil);
             END;
          = seqname:
             BEGIN
             astruc(&astrng);
             nhash _ hash(&astrng);
             ptr[1] _ 1;
             ptr _ getnxt(ptr);
             END;
          = braname:
             BEGIN
             ptr[1] _ 1;
             RETURN(ptr _ namingrp(ptr, ptr, &astrng, 1000));
             END;
          = sid:
             RETURN(ptr _ lkupfast(ptr, 0, &astrng, rsid));
          ENDCASE IF ptr[1]=empty THEN ptr[1]_1;
       UNTIL ptr = endfil DO
          BEGIN
          IF inptrf THEN %have a rubout%
             BEGIN
             ptr _ endfil;
             RETURN;
             END;
          CASE type OF
             =seqname:
                IF getnam(ptr) = nhash THEN
                   BEGIN
                   CCPOS SF(ptr);
                   <UTILTY, xtrnam>($lkupreg, $swork, -1, 0);
                   IF <UTILTY, compas>($lkupreg, &astrng) THEN
                   RETURN;
                   END;
             =contnt:
                IF FIND ptr > [*astrng*] ^ptr _ptr THEN RETURN;
             =cont1s: %search is limited to a single statement%
                BEGIN
                IF NOT FIND ptr > [*astrng*] ^ptr _ptr THEN ptr _
                endfil;
                RETURN;
                END;
             =wordtyp:
                BEGIN
                CCPOS ptr;
                LOOP
                   IF FIND [*astrng*] ^ptr _ptr < THEN
                      BEGIN
                      count _ astrng.L;
                      UNTIL (count _ count-1) < empty DO FIND CH;
                      IF FIND ^p1scn -LD AND ptr > CH -LD THEN
```

```
                    RETURN;
                    IF NOT FIND > plscn CH THEN EXIT;
                       % this can EXIT if astrng is empty and
                       have reached end of statement %
                    END
                 ELSE EXIT;
              END;
           =wordls:
              BEGIN
              CCPOS ptr;
              LOOP
                 IF FIND [*astrng*] ^ptr _ptr < THEN
                    BEGIN
                    count _ astrng.L;
                    UNTIL (count _ count-1) < empty DO FIND CH;
                    IF FIND ^plscn -LD AND ptr > CH -LD THEN
                    RETURN;
                    IF NOT FIND > plscn CH THEN
                       % this can RETURN if astrng is empty and
                       have reached end of statement %
                       BEGIN
                       ptr _ endfil;
                       RETURN;
                       END;
                    END
                 ELSE
                    BEGIN
                    ptr _ endfil;
                    RETURN;
                    END;
              END;
           ENDCASE err($"NLS system error");
        ptr _ getnxt(ptr);
        ptr[1] _ 1;
        END;
     RETURN
     END.

     %  %
```

```
(specreg)                                                    3B9C
   %Spec a register.  This procedure converts a string or a
   statement identifier to a t-pointer.  The conversion
   algorithm depends on the first character in the register,
   and on the parameter passed as the second argument,  If a
   string is being matched, the routine expects as a third
   argument the stid at which the search should begin,
      If the first character is a number, and a name is being
      speced, the register is assumed to contain a statement
      number, and FECHUX is used to convert the A-string to a
      STID.  Otherwise the register is assumed to contain
      either a word or a string to be matched.
      If TYPE (the second argument) is
      1, the register is assumed to contain a name;
      2, the register is assumed to contain a word;
      3 or 6, the register is assumed to contain a string;
      4, the register is assumed to contain a statement
      identifier;
      In all of these cases LOOKUP is called to find the
      STID.%
   %---------------%
   PROCEDURE(astrng, type, ptr);
   REF astrng, ptr;
   IF astrng.L = empty
      THEN
         BEGIN
         ptr.stpsid _ origin;
         ptr[1] _ 1;
         RETURN;
         END;
   IF type = nametyp OR type = seqname OR type = nxtname OR
   type = extname
      THEN
         BEGIN %number, sid or name%
         ptr[1] _ 1;
         IF *astrng*[ 1 ] IN [*0, *9]
            THEN
               BEGIN %number or SID%
               IF *astrng*[ 1 ] = *0 AND astrng.L > 1  % sid
               given %
                  THEN
                     BEGIN
                     *astrng* _ *astrng*[2 TO astrng.L];
                     %delete *0 %
                     lookup(&ptr, cvsno(&astrng), sid);
                     %lookup sid%
                     RETURN;
                     END;
               ptr _ fechux(&astrng, ptr.stfile);
               RETURN;
               END;
         END;
   lookup(&ptr, &astrng, type);
   RETURN;
   END.
```

% %

% %

```
(cvsno) PROCEDURE(astr); %convert a-string to number%      3B9E
   %An A-string containing digits can be converted to a binary
   integer by calling CVSNO with the A-string address.
   Theconversion is done to the base 10, and all characters
   are assumed to be digits, and are not checked.  The
   A-string is not changed, and the integer is returned.  If
   the first character is a  minus sign, the number will be
   converted to a negative number correctly.%
   %----------------%
   LOCAL
      chrcnt, %character count for char readout%
      negnum, %negative number flag%
      number; %cell in which the number is constructed%
   REF astr;
   number _ negnum _ 0;
   chrcnt _ empty + 1;
   IF *astr*[chrcnt] = '- THEN BUMP chrcnt, negnum;
   DO number _ number*10 + (*astr*[chrcnt] - '0)
   UNTIL (chrcnt := chrcnt + 1) = astr.L;
   RETURN(IF negnum THEN -number ELSE number);
   END.

(lkupfast)PROC(stid, start, value, field);                 3B9F
   LOCAL rnb, rnptr, rnt, pgindx, rngbkend, rngbkptr, fileno;
   REF rnptr, rngbkptr;
   fileno _ stid.stfile;
   rnt _ (rnb _ filehead[fileno]+$rngst-$filhed) + rngm;
   &rnptr _ rnb+start.stblk-1;
   IF start THEN
      BEGIN
      IF (start _ start.stwc + ringl) >= blksiz THEN  %Gone
      over to te next block%
         BEGIN
         BUMP &rnptr;
         start _ 0;
         END
      ELSE start _ start - fbhdl;
      END;
   WHILE (&rnptr _ &rnptr+1) < rnt DO
      IF rnptr.rfexis THEN
         BEGIN
         IF (pgindx _ rnptr.rfcore) = 0 THEN
            pgindx _ lodrfb(&rnptr-rnb+rngbas, rngtyp,
            fileno);
         rngbkend _ (&rngbkptr _ crpgad[pgindx] + fbhd]) +
         blksiz;
         IF start THEN &rngbkptr _ &rngbkptr + start := 0;
         %add  in displacement if it's there%
         WHILE &rngbkptr < rngbkend DO
            IF rngbkptr.field = value THEN
               BEGIN
               stid.stblk _ &rnptr-rnb;
               stid.stwc _ &rngbkptr-crpgad[pgindx];
               RETURN(stid);
               END
            ELSE &rngbkptr _ &rngbkptr + ringl;
```

```
        END;
    RETURN(endfil);
    END.

    %  %
```

```
(namingrp) %lookup name in file starting at stid and
terminating when through with the branch headed by lbstid,
looking down levels levels.%                                    3B9G
   % This procedure finds the statement whose name is in
   namstr.  the search is restricted to that part of the file
   begining with STID through the branch headed by LBSTID, and
   within LEVELS levels below STID. %
   %-------------------%
   PROC(stid, lbstid, namstr, levels);
   LOCAL rng, nhash, save;
   LOCAL STRING locstr[100];
   REF namstr, rng;
   IF stid = endfil THEN RETURN(endfil);
   astruc(&namstr);
   nhash _ hash(&namstr);
   save _ rubabt := FALSE;
   LOOP
      BEGIN
      IF getnmf(stid) AND getnam(stid) = nhash THEN
         BEGIN
         CCPOS SF(stid);
         xtrnam($locstr, $swork, -1, 0);
         IF *namstr* = *locstr* THEN
            BEGIN
            rubabt _ save;
            RETURN(stid);
            END;
         END;
      IF levels > 0 AND (stid := getsub(stid)) # stid THEN
         BUMP DOWN levels
      ELSE
         LOOP
            IF getftl(stid) THEN
               BEGIN
               BUMP levels;
               IF stid = lbstid OR inptrf THEN EXIT LOOP 2;
               stid _ getsuc(stid); %GETSUC RETURNS THE UP IF
               TAIL%
               END
            ELSE
               BEGIN
               IF stid = lbstid OR inptrf OR stid.stpsid =
               origin THEN
                  EXIT LOOP 2;
               stid _ getsuc(stid);
               EXIT LOOP;
               END;
      END;
   rubabt _ save;
   RETURN(endfil);
   END.

   %  %
```

(bela)	<nine, auxcod, 02415>	EXT CONSTANT =2	3A11
(belb)	<nine, auxcod, 02414>	EXT CONSTANT =1	3A10
(belba)	<nine, auxcod, 02416>	EXT CONSTANT =3	3A12
(belno)	<nine, auxcod, 02413>	EXT CONSTANT =0	3A9
(cetcapp)	<nine, auxcod, 02316>	PROCEDURE	13C
(changcom)	<nine, auxcod, 0495>	PROC	7A
(clrwa)	<nine, auxcod, 02338>	EXT CONSTANT =2	3A3
(clrwb)	<nine, auxcod, 02337>	EXT CONSTANT =1	3A2
(clrwba)	<nine, auxcod, 02411>	EXT CONSTANT =3	3A4
(clrwno)	<nine, auxcod, 02336>	EXT CONSTANT =0	3A1
(cnvcrifteol)	<nine, auxcod, 01947>	PROCEDURE	13A
(cnveoltcrlf)	<nine, auxcod, 02307>	PROCEDURE	13B
(cra)	<nine, auxcod, 02343>	EXT CONSTANT =2	3A7
(crb)	<nine, auxcod, 02342>	EXT CONSTANT =1	3A6
(crba)	<nine, auxcod, 02412>	EXT CONSTANT =3	3A8
(crlf)	<nine, auxcod, 01770>	PROCEDURE	4I
(crno)	<nine, auxcod, 02341>	EXT CONSTANT =0	3A5
(daofrmt)	<nine, auxcod, 01573>	LOCAL	9C
(datfrmt)	<nine, auxcod, 02247>	LOCAL	9D
(delmkn)	<nine, auxcod, 0509>	PROCEDURE	8A
(delmkr)	<nine, auxcod, 0535>	PROCEDURE	8B
(diserr)	<nine, auxcod, 01341>	PROCEDURE	4D
(dismes)	<nine, auxcod, 02060>	PROCEDURE	4A
(dtfrmt)	<nine, auxcod, 01566>	LOCAL	9B
(febela)	<nine, auxcod, 02492>	EXT CONSTANT =20B	3A17
(febelb)	<nine, auxcod, 02493>	EXT CONSTANT =40B	3A18
(fecames)	<nine, auxcod, 02495>	EXT CONSTANT =200B	3A20
(fecont)	<nine, auxcod, 02494>	EXT CONSTANT =100B	3A19
(fecra)	<nine, auxcod, 02488>	EXT CONSTANT =1B	3A13
(fecrb)	<nine, auxcod, 02489>	EXT CONSTANT =2B	3A14
(fecwa)	<nine, auxcod, 02490>	EXT CONSTANT =4B	3A15
(fecwb)	<nine, auxcod, 02491>	EXT CONSTANT =10B	3A16
(fstatus)	<nine, auxcod, 0726>	PROCEDURE	11A
(getdat)	<nine, auxcod, 01560>	LOCAL	9A
(gmtfset)	<nine, auxcod, 02239>	PROCEDURE	9G
(gofork)	<nine, auxcod, 01783>	PROCEDURE	5K
(goroot)	<nine, auxcod, 0168>	PROCEDURE	5A
(gps)	<nine, auxcod, 0171>	PROCEDURE	5B
(gtadcall)	<nine, auxcod, 02217>	PROCEDURE	9E
(halt)	<nine, auxcod, 0221>	PROCEDURE	5I
(idfrmt)	<nine, auxcod, 01594>	LOCAL	9H
(insmkr)	<nine, auxcod, 0554>	PROCEDURE	8C
(iodaterr)	<nine, auxcod, 01330>	LOCAL	5E1
(ioderr)	<nine, auxcod, 0194>	LOCAL	5E5
(iodtproc)	<nine, auxcod, 0190>	PROC	5E
(jctlcres)	<nine, auxcod, 0429>	PROC	6H
(joctlc)	<nine, auxcod, 0417>	PROC	6G
(lockpage)	<nine, auxcod, 0827>	PROCEDURE	12B
(msgpsi)	<nine, auxcod, 0134>	PROCEDURE	4H
(nlsrst)	<nine, auxcod, 0174>	PROCEDURE	5C
(nmdlgset)	<nine, auxcod, 0704>	PROCEDURE	10C
(nmdlsset)	<nine, auxcod, 0650>	PROCEDURE	10A
(pause)	<nine, auxcod, 0216>	PROCEDURE	5H
(plxset)	<nine, auxcod, 0711>	LOCAL	10D
(psi)	<nine, auxcod, 01319>	PROCEDURE	6I
(rerror)	<nine, auxcod, 0178>	PROCEDURE	5D

< NINE, AUXCOD.NLS;24, >, 30-May-78 16:36 HGL ;;;;

```
FILE auxcod % <ARCSUBSYS>L109 to <RELNINE>auxcod % % (arcsubsys,L109,)
(RELNINE,auxcod.rel,) %

ALLOW!
%.....declarations.....%

   % constants for clearing window, CR and bell control in displaying
   messages %
      (clrwno) EXTERNAL CONSTANT = 0; %do not clear window%          3A1
      (clrwb) EXTERNAL CONSTANT = 1;  %clear window before msg display%
                                                                     3A2
      (clrwa) EXTERNAL CONSTANT = 2;  %clear window after msg display%
                                                                     3A3
      (clrwba) EXTERNAL CONSTANT = 3;  %clear window before and after
      msg display%                                                   3A4
      (crno) EXTERNAL CONSTANT = 0; %no CR%                          3A5
      (crb) EXTERNAL CONSTANT = 1;  %CR before msg display%          3A6
      (cra) EXTERNAL CONSTANT = 2;  %CR after msg display%           3A7
      (crba) EXTERNAL CONSTANT = 3;  %CR before and after msg display%
                                                                     3A8
      (belno) EXTERNAL CONSTANT = 0; %no bell%                       3A9
      (belb) EXTERNAL CONSTANT = 1;  %bell before msg display%      3A10
      (bela) EXTERNAL CONSTANT = 2;  %bell after msg display%       3A11
      (belba) EXTERNAL CONSTANT = 3;  %bell before and after msg
      display%                                                      3A12
      (fecra) EXTERNAL CONSTANT = 1B;                               3A13
      (fecrb) EXTERNAL CONSTANT = 2B;                               3A14
      (fecwa) EXTERNAL CONSTANT = 4B;                               3A15
      (fecwb) EXTERNAL CONSTANT = 10B;                              3A16
      (febela) EXTERNAL CONSTANT = 20B;                             3A17
      (febelb) EXTERNAL CONSTANT = 40B;                             3A18
      (feconf) EXTERNAL CONSTANT = 100B;                            3A19
      (fecames) EXTERNAL CONSTANT = 200B;                           3A20
   (cttywindow) EXTERNAL CONSTANT = 1002; % tty window flag %
   (cctbwindow) EXTERNAL CONSTANT = 1003; %  cmd feedback window flag %
   REF rawchr, msgda, tda;
   EXTERNAL iodaterr;
   EXTERNAL ilspsi;
   EXTERNAL stopline;
   EXTERNAL rubout;
%...Message Display...%
   (dismes)   % CL:GB ;  display a message in the user's tty window %
   PROCEDURE (type, astrng REF % =>  %);                             4A
      % Procedure description
         FUNCTION
            This routine is called to display a message in the user's
            tty window.  Two LF'S will be inserted before message so
            that previous message will disappear.  It calls typseq.
            Checks Disable Status flag and returns immediately if flag
            is true.
         ARGUMENTS
            type -- INTEGER - value determines the display action
               type = 0:
                  will remove any message on the screen. An A-string
```

```
            need not be given in astrng in this case.
         type > 0:
            the message will be displayed, and the routine will
            return (with the message still on the screen).
      astrng -- REF - addr of string to be displayed, if type is
      non-zero.
   RESULTS
      none
   NON-STANDARD CONTROL
   GLOBALS
      none
   %
% Declarations %
% check disable show status flag %
   IF dssflag THEN RETURN;
% Check for clear window and call typseq %
   IF type = 0 THEN  %just clear window.  send empty string%
      typseq(cttywindow, nocawait, clrwb, crno, belno, $"")
   ELSE
      BEGIN
      typseq(cttywindow, nocawait, clrwb, crno, belno, &astrng);
      END;
% return %
   RETURN;
END.

(typelit)   % CL:GB ;  display a message in the user's command
teedback window %
PROCEDURE (type, astrng REF % =>  %);                              4B
   % Procedure description
      FUNCTION
         This routine displays a message in the user's commad
         feedback window and optionally requires user's confirmation
         to continue.  It calls typseq to cause FE to display the
         string.  No clearing of window or appending of CR before or
         after.
      ARGUMENTS
         type -- INTEGER - value determines the display action
            type = 0, nocawait:
               will return control after displaying message.
            type = 1, cawait:
               if display terminal, require user confirmation before
               returning control.  Append "Type OK:" to user
               message.
            type = 2, cawtnomes:
               if display terminal, require user confirmation before
               returning control.  No message appended.
         astrng -- REF - addr of string to be displayed.
      RESULTS
         none
      NON-STANDARD CONTROL
      GLOBALS
      %
   % Declarations %
   % call typseq  %
      typseq(ccfbwindow, IF type NOT IN [nocawait,cawtnomes] THEN
```

```
      cawait ELSE type, clrwno, crno, belno, &astrng);
   % return %
      RETURN;
   END.

(typseq)   % CL:GB ;  display a message to user %
PROCEDURE (disptype, dispaction, clearw, crcntrl, belcntrl, astrng
REF % => %);                                                    4C
   % Procedure description
      FUNCTION
         This routine calls the Frontend WRIT-SEQ to display the
         message in the appropriate window.
      ARGUMENTS
         DISPTYPE:  Window in which the message is to be displayed.
                                                                4C1B1
            The most common values for this parameter will be the
            generic window IDs:                                 4C1B1A
            = cttywindow (1002): TTY window
            = ccfbwindow (1003):  Command feedback window.
            Other valid window IDs for specific windows may be used.
                                                                4C1B1D
         DISPACTION:  Confirmation control.                      4C1B2
            = nocawait (0):  Don't wait for confirmation from the
            user.
            = cawait (1):  Wait for confirmation from the user.
            Append the message "Type OK:" to the string pointed to
            by the sixth parameter.
            = cawtnomes (2):  Wait for confirmation.  No appended
            message.
         CLEARW:  Window clearing control.                       4C1B3
            = clrwno (0): do not clear window.
            = clrwb (1): clear window before.
            = clrwa (2): clear window after.
            = clrwba (3): clear window before and after.
         CRCNTRL:  Carriage return control.                      4C1B4
            = crno (0): no CRLF.
            = crb (1): CRLF before.
            = cra (2): CRLF after.
            = crba (3): CRLF before and after.
         BELCNTRL:  Bell control.                                4C1B5
            = belno (0): no bell.
            = belb (1): bell before.
            = bela (2): bell after.
            = belba (3): bell before and after.
         ASTRNG - REF:  Address of string to be displayed.       4C1B6
      RESULTS
         none
      NON-STANDARD CONTROL
         SIGNALS GENERATED
            none
      GLOBALS
         dsparg:  global list; sets and nulls out
      %
   % Declarations%
      LOCAL
         firstline _ TRUE,  % set to FALSE after firstline %
```

```
        wrkstr REF,  % addr of current substring %
        chsent = 0,  % characters sent %
        strsize,  % size of astrng %
        maxwrds = maxdpschar/5,  % max words that can be sent to FE
        %
        savel,  % save word for header replacement %
        wrdptr = 0,  % pointer to header of next substring %
        windcontrol, % used to build window control argument to the
        FE %
        notdone = TRUE;
    REF dsparg;
% package and send the string, within FE constraints %
    &wrkstr _ &astrng;
    strsize _ astrng.L;
    % save the header %
        savel _ astrng;
    INVOKE (cattypseq);
    WHILE notdone DO
        BEGIN
        IF wrkstr.L > maxdpschar THEN
            BEGIN
            % send full words only, unless last string %
                wrkstr.L _ maxdpschar;
            % fixup charcnt and wrdcnt %
                wrdptr _ wrdptr + maxwrds;
                chsent _ chsent + maxdpschar;
            END
        ELSE notdone _ FALSE;
        % Build argument list for FE WRIT-SEQ procedure %
            windcontrol _ 0;
            IF firstline := FALSE THEN
                BEGIN
                CASE crcntrl OF
                    = crb, = crba: windcontrol _ windcontrol .V fecrb;
                    ENDCASE;
                CASE clearw OF
                    = clrwb, = clrwba: windcontrol _ windcontrol .V
                    fecwb;
                    ENDCASE;
                CASE belcntrl OF
                    = belb, = belba: windcontrol _ windcontrol .V
                    febelb;
                    ENDCASE;
                END;
            IF NOT notdone THEN % last portrayal of message %
                BEGIN
                CASE crcntrl OF
                    = cra, = crba: windcontrol _ windcontrol .V fecra;
                    ENDCASE;
                CASE clearw OF
                    = clrwa, = clrwba: windcontrol _ windcontrol .V
                    fecwa;
                    ENDCASE;
                CASE belcntrl OF
                    = bela, = belba: windcontrol _ windcontrol .V
                    febela;
```

```
                ENDCASE;
             windcontrol _
                (CASE dispaction OF
                   = nocawait: 0;
                   = cawait: feconf .V fecames;
                   ENDCASE % cawtnomes % feconf ) .V windcontrol;
             END;
          #dsparg# _
             USE makedesc (uindex, disptype % window handle %,
             FALSE),
             *wrkstr*,
             USE makedesc (uindex, windcontrol, FALSE);
       % send the string %
          IF NOT extcall(feident, %prs hardle or mailbox %
          $"writ-seq", fetoolpkg % package handle or callmode %,
          &dsparg) THEN
             ABORT( erfeshow, $"FE failure while writing text.");
       % move header past last end-of-line %
          IF notdone THEN
             BEGIN
             % restore previous header and first word%
                wrkstr _ save1;
             % save next header and first word %
                save1 _ wrkstr[wrdptr];
             % move work string, set-up header %
                &wrkstr _ &wrkstr + wrdptr;
                wrkstr.L _ wrkstr.M _ strsize - chsent;
             END;
       % null argument list %
          #dsparg# _ ;
       END;
   % return %
      DROP (cattypseq);
      RETURN;
   (cattypseq)  CATCHPHRASE;                                    4C5
      BEGIN
      DISABLE (cattypseq);
      CASE SIGNALTYPE OF
         = notetype:
            CASE SIGNAL OF
               = return, = unwind :
                  BEGIN  %null lists%
                  #dsparg# _ ;
                  END;
            ENDCASE CONTINUE;
      ENDCASE CONTINUE;
      END;
   END.

(diserr)  %display error message to the user%
PROCEDURE (astrng REF);                                         4D
   %--------------%
   dismes (1, &astrng);
   RETURN;
   END.
(specttyout)PROC(ttyno, astr);                                  4E
```

```
   %This procedure accepts a teletype numbr and a string, and types
   the string as a message on the tty, or to jlog file if ttyno=0.
      It additionally typeas a series of attention getting
      characters on the tty.%
   LOCAL STRING error[100];
   LOCAL ttjfn, ttlocjfn;
   REF astr;
   ttlocjfn _ 0;
   IF jdebug THEN RETURN; %Don't do it if debugging%
   IF ttyno=0 THEN
      BEGIN
      IF jlogjfn=0 THEN
         BEGIN
         IF NOT ttlocjfn _ sgtjfn (0, $"<JOURNAL>JLOG.TXT;P777724",
         $error) THEN
            BEGIN
            *error* _ "Can't get JFN for JLOG file:  ", *error*;
            specttyout (oprtty,$error);
            RETURN;
            END;
         IF NOT sysopen (ttlocjfn, append, chrtyp, $error) THEN
            BEGIN
            IF NOT SKIP !rljfn(ttlocjfn) THEN NULL;
            *error* _ "Trying to log journal error:  ", *error*;
            specttyout (oprtty,$error);
            RETURN;
            END;
         ttjfn_ttlocjfn;
         END
      ELSE ttjfn_100B;
      END
   ELSE
      BEGIN
      %Check that tty is available%
         R1 _ getsbn($"TTYJOB");
         !JSYS sysgt;
         IF (R1.RH _ R2.RH) = 0 THEN RETURN; %Table not thee%
         R1.LH _ ttyno;
         IF (NOT SKIP !JSYS getab) OR (R1 # -1) THEN RETURN;
      %set ttjfn for tty%
         ttjfn_ 485 .V ttyno;
      END;
   %First the attention getters%
      !sout(ttjfn, chbptr(0) + $"_____--", -7);
   %Now the string%
      !sout(ttjfn, chbptr(0) + &astr, -astr.L);
   %Now another EOL%
      !bout(ttjfn, EOL);
   IF ttlocjfn AND NOT sysclose(ttlocjfn, $error) THEN dismes(1,
   $error);
   RETURN
   END.
(settimer)  %set the system timer -- after MILLISECONDS
milliseconds, PROC will be called with ARG1 thru ARG4%
PROCEDURE (milliseconds, proc, arg1, arg2, arg3, arg4);          4F
   LOCAL frkac0, frkac1;
```

```
   stoptimer();
   timrset _ TRUE;
   timrproc _ proc;
   timra1 _ arg1;
   timra2 _ arg2;
   timra3 _ arg3;
   timra4 _ arg4;
   frkac1 _ MIN(10000, milliseconds); %milliseconds to wait%
   !sfacs(msgfrk, $frkac0); %set timer fork acs%
   !stork(msgfrk, 120B); %start the timer fork%
   !rfork(); %thaw fork, r1 already set%
   RETURN;
   END.

(stoptimer)  %stop the system timer%
PROCEDURE;                                                          4G
   %should be able to simply halt the fork, but 10X calls it an
   error to halt a fork which is already halted, STUPID STUPID%
   !ffork(msgfrk); %freeze the fork%
   !rfsts(msgfrk); %read status%
   IF NOT R1 .A 2B6 THEN
      !hfork(msgfrk); %halt the fork%
   timrset _ FALSE;
   RETURN;
   END.

(msgpsi) %pseudo interrupt from the timer fork%
PROCEDURE;                                                          4H
   svac1 _ R1;
   R1 _ $svacs;
   !BLT R1, svacse;
   S _ S + 40000040B;
   timrset _ FALSE;
   %call the designated procedure%
      [timrproc](timra1, timra2, timra3, timra4);
   R1.LH _ $svacs;
   R1.RH _ 0;
   !BLT R1, 17B;
   R1 _ svac1;
   !JSYS debrk;
   END.

(crlf) PROCEDURE; %type a carriage return-line feed%                4I
   %-----------------%
   !pbout(CR);
   !pbout(LF);
   RETURN;
   END.

(typeas) PROCEDURE (astrng); %Type a-string on tty%                 4J
   %-----------------%
   REF astrng;
   IF NOT astrng.L THEN RETURN;
   !sout(101B, chbmty + &astrng, -astrng.L);
   RETURN;
   END.
```

```
%.....error, abort, and termination routines.....%
   (goroot) PROCEDURE;  %Goto root%                                    5A
      ABORT(statesig, 0);
      END.
   (gps) PROCEDURE;  %GOTO STATE routine%                              5B
      ABORT(statesig, 0);
      END.

   (nlsrst) PROCEDURE;  %RESET NLS to rcover from castastrophic
   problems%                                                           5C
      %Clean up the Backend and go back to the Frontend%
      halt();
      END.

   (rerror) PROCEDURE;                                                 5D
      LOCAL STRING temp[70];
      dismes(1, $"Fatal error"); %leave for 1 second%
      ddgtsymbol($temp, CM .A 18M] .A 18M -1);
      *temp* _ "Crash at PROCEDURE*  ", *temp*;
      dismes (1, $temp);
      (errhlt):                                                        5D6
         !JSYS 147B; %Reset jsys-- cannot use symbol because of
         conflict%
         !JSYS haltf; GOTO errhlt;
         %In case someone is foolish enough to try to contnue%
      END.

   (iodtproc)PROC;                                                     5E
      (iodaterr):                                                      5E1
      %Come here on an IO data Error (channel 11)%
      [!evtab] _ $ioderr;
      !JSYS debrk;
      (ioderr):                                                        5E5
         ABORT(-6, $"I/O Data Error");
         %We don't know what file the error was on, so for now just
         type a nasty mesage and signal.  deferr will rerror%
      END.
   (thWrtp) PROCEDURE; %thaw random file pages%                        5F
      LOCAL end, ct;
      REF ct;
      end _ (&ct _ $corpst + 1) + rfpmax;
      DO IF ct.ctpnum # 0 THEN ct.ctfroz _ 0
      UNTIL (&ct _ &ct + 1) = end;
      RETURN;
      END.

   (thwfil) PROC(filno); %thaw file pages of file%                     5G
      LOCAL end, ct;
      REF ct;
      end _ (&ct _ $corpst + 1) + rfpmax;
      DO IF ct.ctfile = filno AND ct.ctpnum # 0 THEN ct.ctfroz _ 0
      UNTIL (&ct _ &ct + 1) = end;
      RETURN;
      END.
```

```
(pause) PROCEDURE(tim);                                                 5H
   % pause for specified number of msec %
   R1 _ tim;
   !JSYS 167B; %dismss%
   RETURN
   END.

(halt) PROCEDURE; %terminate NLS%                                       5I
   %close work station (if NLS), set continue, and issue
   terminate jsys%
   %--------------%
   % IF nlmode NOT= typewriter THEN shutdis();%
   % ctlquit();%
   continue _ TRUE;
   closeall();
   LOOP !JSYS haltf;
   % GOTO rentr;%   % if a person forgets and does a CONTINUE rather
   than a RENTER after an Execute Quit, then this will do the RENTER
   for him %
   END.

(shutdis) PROCEDURE; %shut down display before leave NLS%               5J
   REF dspcmd;
   clearda( 0, &dspcmd ); % clear entire window %
   clrall(0, TRUE );
   prcmds( 0, &dspcmd);
   RETURN
   END.

(gofork) PROCEDURE(prcnam, injfn, outjfn); %goto lower fork%            5K
   %run a lower level fork with the specified process, input jfn and
   output jfn, leave ntiw psi on%
   LOCAL
      savnm,  %saved name from tenex%
      pjfn, %jfn for process%
      pfork;  %fork handle for process%
   % get and remember tenex subsystem name %
      !getnm();
      savnm _ R1;
   %create fork%
      R1 _ 2B11; %pass capabilites%
      IF NOT SKIP !JSYS 152B %cfork% THEN
         err($"Can't create fork");
      pfork _ R1;
   %enable capabilities%
      R1 _ pfork;
      R2 _ -1; %enable all%
      R3 _ -1; %enable all%
      !JSYS 151B; %epcap%
   %set primary input if given jfn%
      IF injfn # -1 OR outjfn # -1 THEN BEGIN
         R1 _ pfork;
         R2.LH _ injfn;
         R2.RH _ outjfn;
         !JSYS 207B; %spjfn%
         END;
```

```
    %get process into the new fork%
       IF NOT (pjfn _ lgetjfn($subdir,prcnam,$savext,gtjprf,$lit))
          THEN err($"Can't get jfn for process");
       R1.LH _ pfork;
       R1.RH _ pjfn;
       !JSYS get;
    %start fork using entry vector%
       R1 _ pfork;
       R2 _ 0;
       !JSYS 201B; %sfrkv%
    %wait for process to terminate%
       R1 _ pfork;
       !JSYS 163B; %wfork%
    %kill fork%
       R1 _ pfork;
       !JSYS 153B;  %kfork%
    %release process jfn%
       reljfn(pjfn);
    %restore subsystem name%
       !setnm( savnm );
    %restore display%
       dpset(dspallf, endfil, endfil, endfil);
    %return%
       RETURN
       END.

%...Pseudo-interrupt routines...%
   (rubproc) PROCEDURE; %rubout interupt routine%                    6A
      %----------------------%

      %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%

      %----------------------%
      (rubout):                                                      6A6
      %save registers%
      svac1 _ R1;
      R1 _ $svacs;
      !BLT R1, svacse;
      IF iibflg THEN killib();
      IF rubabt:= FALSE THEN [levtab] _ $rabort
      ELSE [rubmrk] _ TRUE;
      %restore registers%
      !HRLZI R1, svacs;
      !BLT R1, 17B;
      R1 _ svac1;
      !debrk();
      (rabort): %come here for real aborts%                          6A19
         rubsig();
      END.

   (rubsig) PROC;                                                    6B
      %simply calls signal for rubout%
      ABORT(statesig, 0);
      END.

   (stoppproc) PROCEDURE; % ^S interrupt routine%                    6C
```

```
   %----------------------%

   %INTERRUPT ROUTINE==NO LOCALS OR SUBROUTINE CALLS%

   %----------------------%
   (stopline):                                                    6C6
   %set flag%
   !AOS inpstp;
   !JSYS debrk;
   END.

(trapc)  PROCEDURE; %no-op control c%                             6D
   !SOSLE ccignore;
   !JSYS debrk;
   GOTO [savchntab[2]];
   END.

(trapo)  PROCEDURE; %no-op control o%                             6E
   !SOSLE ccignore;
   !JSYS debrk;
   GOTO [savchntab[3]];
   END.

(traps)  PROCEDURE; %no-op control S%                             6F
   !SOSLE ccignore;
   !JSYS debrk;
   GOTO [savchntab[1]];
   END.

(joctlc)PROC;                                                     6G
   %Come here on a control c%
   !JSYS 141B; %cis--clear intterupt system (can't us set because of
   duplicate symbol)%
   jclosfl();
   jctlcres();
   R1 _ -1; %exec%
   R2 _ 2B11; %^C channel%
   !JSYS iic; %cause interrupt%
   R1 _ 2000; !JSYS dismS; %for TENEX timing problem%
   dismes(2,$"Journal Process Aborted by ^C");
   nisrst();
   END.

(jctlcres)PROC;                                                   6H
   IF NOT jsavaccess THEN disabaccess(0, jrnlaccess);
   %connect back to original directory%
      conjdir(FALSE);
   %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;
    RETURN;
    END.

(psi) PROCEDURE;  %illegal instruction interrupt routine%          6I
    (ilspsi):                                                      6I1
        !SKIPE ilsdsp;
        !JRST @ilsdsp;
    % default routine - save relevant information and !debrk  %
        (ilsdefault):                                              6I2A
        % save all 16 registers %
            !MOVEM 0,psireg;
            !HRRI 0,psireg;
            !AOS 0;
            !HRLI 0,1;
            !BLT 0,psirglast;
        % Save illegal instruction and its address %
            BUMP DOWN [levtab];  % get the right instruction %
            ilsins _ [[levtab]];
            ilsadd _ [levtab] .A 777777B;
        % change debrk address to error routine %
            [levtab] _ $psitrm;
            !debrk();
        (psitrm):  % set up error string and ABORT %               6I2F
            % message gets illegal instruction, its address and
            important registers %
                *ilstxt* _  "Illegal instruction.  Backend is dead.
                Instruction= ",
                    STRING ( ilsins, 8),
                    " at ",STRING (ilsadd, 8),
                    " R1 = ", STRING (psireg[1], 8),
                    " R2 = ", STRING (psireg[2], 8),
                    " R3 = ", STRING (psireg[3], 8),
                    " M = ", STRING (psireg[16B], 8),
                    " S = ", STRING (psireg[17B], 8);
            ABORT ( badinstruction, $ilstxt);
    END.
(rstcntlo)   % CL: ;  reset abort output ( <^O> ) %
PROCEDURE;                                                         6J
    % Procedure description
        FUNCTION
            Call FE externally callable routine to resume normal output
            following a control-O.
        ARGUMENTS
            none
        RESULTS
            proc-value
        NON-STANDARD CONTROL
            none
        GLOBALS
            none
        %
    % Declarations %
        LOCAL LIST nullargs[1];
    % reset <^O> flag %
```

```
        inptrf _ FALSE;
      % null argument list %
        #nullargs# _ ;
      % make remote call %
        IF NOT extcall(feident, %process handle or mailbox%
        $"reset-cntl-o", fetoolpkg %package handle or callmode%,
        $nullargs) THEN
          ABORT( conto, $"FE Reset control-O failure.");
      % Return %
        RETURN;
      END.

%.....help routines.....%
   (changcom)PROC(string, retparm);                                    7A
      %This procedure accepts a string as a parameter, and types out a
      message (dismes) saying tht th command has been changed, and that
      the user should consult Folklore for details%
      %Returns in the same manner as help%
      LOCAL count;
      LOCAL STRING msgstring[250];
      REF string;
      IF (count _ string.L) + 50 >msgstring.M THEN count _ 200;
      *msgstring* _ EOL, *string*[1 TO count], EOL, "Command
      Changed--see (documentation, folklore,)";
      dismes(2, $msgstring);
      IF retparm = -1 THEN RETURN;
      ABORT(statesig, 0);
      END.
%.....marker code.....%
   (delmkn) PROCEDURE (fileno, name);                                  8A
      %delete the marker whose name is passed from the specified
      file%
      %------------------%
      LOCAL marker, end, flhd, mkrcount, stid, aring;
      REF marker, mkrcount, aring;
      % make sure we have a locked file %
        stid _ origin;
        stid.stfile _ fileno;
        lodent( stid, rngtyp : &aring );
        aring.rsub _ aring.rsub;
      find _ filhdr(fileno) - $filhed;
        %subtract $filhed here instead of throughout procedure%
      &marker _ flhd + $mkrtb;
      end _ &marker + [&mkrcount _ flhd + $mkrtbl]*mkrl;
      LOOP
        BEGIN
        IF &marker >= end THEN RETURN; %no such marker%
        IF marker.mkname = name THEN EXIT;
        &marker _ &marker + mkrl;
        END;
      mvbfbf(&marker+mkrl, &marker, end-&marker-mkrl);
      mkrcount _ mkrcount - 1;
      RETURN;
      END.

   (delmkr) PROCEDURE (tptrs);                                         8B
```

```
    %delete the markers in T-string specified by arg%
    LOCAL marker, end, flhd, mkrcount;
    REF marker, mkrcount;
    flnd _ filhdr([tptrs].stfile) - $filhed;
    &marker _ flhd + $mkrtb;
    end _ &marker + [&mkrcount _ flhd + $mkrtbl]*mkrl;
    UNTIL &marker >= end DO
       IF marker.mkpsid = [tptrs].stpsid AND
       marker.mkccnt IN [[tptrs+1],[tptrs+3]) THEN
          BEGIN
          mvbfbf( &marker+mkrl, &marker, end-&marker-mkrl);
          mkrcount _ mkrcount - 1;
          end _ end - mkrl;
          END
       ELSE &marker _ &marker + mkrl;
    RETURN;
    END.

(insmkr) PROCEDURE (bug, name);                                    8C
    %insert marker%
    LOCAL marker, end, flhd, mkrcount, newmkr, mkrmaxlen;
    REF marker, mkrcount, mkrmaxlen;
    flhd _ filhdr([bug].stfile) - $filhed;
    &marker _ flhd + $mkrtb;
    &mkrmaxlen _ flhd + $mkrtxn;
    end _ &marker + [&mkrcount _ flhd + $mkrtbl]*mkrl;
    newmkr _ TRUE;
    UNTIL &marker >= end DO
       BEGIN
       IF marker.mkname = name THEN
          BEGIN
          newmkr _ FALSE;
          EXIT;
          END;
       &marker _ &marker + mkrl;
       END;
    IF newmkr THEN
       BEGIN
       IF mkrcount * mkrl >= mkrmaxlen THEN
          err($"Marker table too long -- new marker not added");
       BUMP mkrcount;
       marker.mkname _ name;
       END;
    marker.mkpsid _ [bug].stpsid;
    marker.mkccnt _ [bug+1];
    RETURN;
    END.

(seemkr) PROCEDURE (fileno, astr);                                 8D
    LOCAL marker, end, flhd, mkrcount, word, stid, count, char;
    LOCAL TEXT POINTER tptr;
    LOCAL STRING locstr[40], lstr2[40];
    REF marker, mkrcount, astr;
    flnd _ filhdr(fileno) - $filhed;
    &marker _ flhd + $mkrtb;
    end _ &marker + [flhd + $mkrtbl]*mkrl;
```

```
    stid _ 0;
    stid.stfile _ fileno;
    *astr* _ NULL;
    UNTIL &marker >= end DO
       BEGIN
       *locstr* _ "   ";
       count _ 0;
       word _ marker.mkname;
       UNTIL (count _ count + 1) = 6 DO
          BEGIN
          char _ CASE count OF
             =1: word.chr0;
             =2: word.chr1;
             =3: word.chr2;
             =4: word.chr3;
             ENDCASE word.chr4;
          IF char # 0 THEN *locstr* _ *locstr*, char
          ELSE *locstr* _ *locstr*, SP;
          END;
       stid.stpsid _ marker.mkpsid;
       % get current location to astr %
          tptr _ stid; tptr[1] _ marker.mkccnt;
          ccurloc( $tptr, $lstr2 );
       *astr* _ *astr*, *locstr*, *lstr2*, EOL;
       &marker _ &marker + mkrl;
       END;
    RETURN;
    END.

%.....time/date routine, ident.....%
   (getdat) %put date and time in astring passed%                    9A
      PROCEDURE(astng);
      dtfrmt(-1, astng); %format current date%
      RETURN;
      END.

   (dtfrmt) %put date and time in canonical form in astring passed%  9B
      PROCEDURE(tim, astng);
      REF astng;
      datfrmt( tim, 201B6, &astng);
      RETURN;
      END.

   (daofrmt) %put date only in canonical form in astring passed%     9C
      PROCEDURE(tim, astng);
      REF astng;
      datfrmt( tim, 601B6, &astng);
      RETURN;
      END.

   (datfrmt) %put date and time in string passed%                    9D
      %tim is internal TENEX format, however TOPS20 format is handled
      since some these got into NLS files during the early days of
      ISID.  The time is put in astng assuming "lcltimzon" as time zone
      (typically 8 for Pacific time).  This leaves the possibility of
      having "lcltimzon" a useroption.%
```

```
   PROCEDURE (tim, frmt, astng);
   LOCAL bytptr, count, savr5;
   REF astng;
   IF tim = -1 THEN tim _ gtadcall();
   IF tim.RH >= 250600B THEN %assume it's an old t20 format still
   nanging around%
      BEGIN
      tim _ tim + 2525258; %assume it had wrong time zone%
      !HRRZ R3,tim;
      !IMULI R3,250600B;
      !ADDI R3,400000B; %rounding%
      !HLRM R3,tim;
      END;
   %now have it in TENEX internal format%
   IF tops20flag THEN %convert to tops20 internal format%
      BEGIN
      !HRRZ R3,tim;
      !IMULI R3,6042710;
      !ADDI R3,100000B;
      !LSH R3,2; %day fraction (*1B6) in R3.LH%
      !HLRM R3,tim;
      END;
   bytptr _ chbptr(astng.L) + &astng;
   !odcnv(0, tim, 0, 1B11+lcltimzon*1B6);
   R1 _ bytptr;
   savr5 _ R5; % because this register is used by compilers, it must
   be saved and restored %
   R5 _ frmt;
   !odtnc();
   R5 _ savr5;
   count _ slngth(bytptr, R1);
   IF astng.L + count > astng.M THEN err($"NLS internal error,
   string too long");
   astng.L _ astng.L + count;
   RETURN;
   END.
(gtadcall)PROCEDURE;                                                    9E
   %get internal date and time in TENEX (right half is seconds since
   GMT midnight) format%
   !gtad();
   IF NOT tops20flag THEN RETURN(R1);
   %the following two statements add 8 hours to time if GMT is the
   local time.  This is because ISID is now (7-JAN-77) running that
   way, but with Pacific time (due to a glitch in BATCH)%
      IF NOT gmtflag THEN gmtfset(); %preserves R1%
      IF gmtflag > 0 THEN R1 _ R1 + 2525258;
         %8 hour correction for having time incorrectly set to GMT%
   !HRRZ R3,R1; %fraction of day * 1B6 in R3.RH%
   !IMULI R3,250600B; %seconds in a day%
   !ADDI R3,400000B; %rounding, seconds since midnight GMT in R3.LH%
   !HLR R1,R3; %put back in R1.RH%
   RETURN (R1);
   END.

(totentad)PROCEDURE; %convert tops20 interal time to tenex%          9F
   %expects tops20 format time in R1, returns tenex format time in
```

```
    R1, smashes R3%
    !HRRZ R3,R1; %fraction of day * 1B6 in R3.RH%
    !IMULI R3,250600B; %seconds in a day%
    !ADDI R3,400000B; %rounding, seconds since midnight GMT in R3.LH%
    !HLR R1,R3; %put back in R1.RH%
    RETURN;
    END.
(gmtfset)PROCEDURE;                                                    9G
    !PUSH S,R1; %preserve R1%
    !odcnv(0,-1,0,0); %find local time zone%
    gmtflag _ IF NOT SKIP !TLNE R4,77B THEN -1 ELSE 1;
       %1 if zone is GMT, -1 if not%
    !POP S,R1;
    RETURN;
    END.
(idfrmt) %put ident into astring passed%                               9H
    PROCEDURE(stid, astng);
    LOCAL sdbadr, word, bytptr, count, char, stdb;
    REF sdbadr, astng;
    % 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 %
       word _ getint(stdb);
       bytptr _ stbptr(empty) + $word;
       count _ 0;
       UNTIL (count _ count + 1) > 4 DO
          IF (char _ ^bytptr) = 0 THEN EXIT LOOP
          ELSE *astng* _ *astng*, char;
    RETURN;
    END.

%.....routines to support name delimiter commands.....%
    (nmdisset) %***% PPOCEDURE  % set name delimiters for a statement %
                                                                       10A
    (stid, dlleft, dlrght);                                            10A1
    LOCAL rnl, sdb;
    LOCAL STRING str[100];
    REF rnl, sdb;
    lodent(stid, rngtyp : &rnl);
    IF NOT lodprop( stid, txttyp : &sdb) THEN
       err($"No text block associated with this node.");
    sdb.slnmdl _ dlleft;
    sdb.srnmdl _ dlrght;
    CCPOS SF(stid);
    *str* _ NULL;
    xtrnam ($str, $swork, dlleft, dlrght);
    IF str.L = empty THEN
       BEGIN
       sdb.sname _ 1;
       rnl.rnameh _ 0;
       rnl.rnamef _ FALSE;
       END
    ELSE
       BEGIN
       sdb.sname _ swork1;
```

```
      astruc($str);
      rnl.rnameh _ hash($str);
      rnl.rnamef _ TRUE;
      END;
   RETURN;
   END.

(xnmdlset) PROCEDURE  % Xecute set name deilimiters for a group % 10B
   (stid1, stid2, dlleft, dlrght, da);                              10B1
   LOCAL sw, stid;
   REF da, sw;
   &sw _ openseq (stid1, stid2, da.davspec, da.davspc2, da.dausqcod,
   da.dacacode);
   UNTIL (stid _ seqgen(&sw)) = endfil DO
      nmdlsset (stid, dlleft, dlrght);
   closeseq (&sw);
   RETURN;
   END.

(nmdlgset) PROCEDURE  % set name delimiters for a group %        10C
   (stid1, stid2, dlleft, dlrght, da);                           10C1
   stid1 _ grptst (stid1, stid2 :stid2);
   xnmdlset (stid1, stid2, dlleft, dlrght, da);
   RETURN;
   END.

(plxset)                                                         10D
   %Given a stid, this routine will return the stid'S of the head
   and tail, respectively, of the plex in which the stid appears.%
   %-------------%
   PROCEDURE(stid);
   LOCAL grp1, %stid of head%
      grp2; %stid of tail%
   IF stid.stpsid = orgstid THEN grp1 _ grp2 _ stid
   ELSE
      BEGIN
      grp1 _ gethed(stid);
      grp2 _ getail(stid);
      END;
   RETURN(grp1, grp2);
   END.

%.....file status routine.....%
   (fstatus) PROCEDURE(fileno, astrng, type);                    11A
      LOCAL fstfdb[25B], drn, nt, numst, fl, cnt, usd, tot, st,
      percent, i, stid;
      LOCAL STRING dname[25];
      REF st, fl, astrng;
      &fl _ flntadr(fileno);
      %file name%
         filnam( fileno, &astrng );
         *astrng* _ *astrng*, EOL;
      %private file?%
         IF rdprvsts (fileno) = $psprivate THEN
            BEGIN
            *astrng* _ *astrng*, "Private File";
```

```
        stid _ 0;
        stid.stpsid _ origin;
        stid.stfile _ fileno;
        IF NOT getfacc (stid, 0, 0, 0) THEN
           *astrng* _ *astrng*, " (but with no Access List)";
        *astrng* _ *astrng*, EOL;
        END;
   % get tdb for Modifications and Size %
      !gtfdb( fl.florig, 25B6, $fstfdb);
        %read entire FDB%
   IF type .A 2 THEN %being modified?%
      BEGIN
      drn _ fstfdb[24B].lkdirn;
      nt _ fstfdb[24B].lkinit;
      IF drn # 0 OR nt # 0 THEN
         <NLS, FILMNP, lockid>(&astrng, drn, nt)
      ELSE *astrng* _ *astrng*, "File not being modified";
      *astrng* _ *astrng*, EOL;
      %browse?%
         IF fl.flbrws THEN *astrng* _ *astrng*, "In temporary
         modifications mode", EOL;
      END;
   IF type .A 4 THEN %directory default for links%
      BEGIN
      IF NOT gdftdir(fileno, $dname) THEN
         *dname* _ "not on this system";
      *astrng* _ *astrng*, "Default directory for links is ",
         *dname*, EOL;
      END;
   IF type = 7 THEN %version creation time%
      BEGIN  %this version%
      *astrng* _ *astrng*, "Creation date of this version: ";
      R1 _ fstfdb[13B];
      IF tops20flag THEN totentad(); %convert R1 to tenext time
      format%
      dtfrmt(R1, &astrng);
      *astrng* _ *astrng*, CR, LF;
      END;
   IF type .A 1 THEN %size%
      BEGIN
      %number of statements%
         numst _ <NLS, VERIFY, crng>(TRUE, fileno);
         *astrng* _ *astrng*, STRING(numst), " statements in file",
         EOL;
      %structure pages%
         &st _ filehead[fileno] + $rngst - $filhed;
         cnt _ 0;
         usd _ tot _ blksiz;
         FOR i _ 0 UP UNTIL = rngm DO
            BEGIN
            IF st.rfexis THEN
               BEGIN
               BUMP cnt;
               usd _ usd + st.rfused;
               tot _ tot + blksiz;
               END;
```

```
            BUMP &st;
            END;
         *astrng* _ *astrng*,
            "Structure pages = ", STRING(cnt), "/", STRING(rngm),
            EOL;
      %data pages%
         &st _ filehead[fileno] + $dtbst - $filhed;
         cnt _ 0;
         FOR i _ 0 UP UNTIL = dtbm DO
            BEGIN
            IF st.rfexis THEN
               BEGIN
               BUMP cnt;
               usd _ usd + st.rfused;
               tot _ tot + blksiz;
               END;
            BUMP &st;
            END;
         *astrng* _ *astrng*,
            "Data pages = ", STRING(cnt), "/", STRING(dtbm), EOL;
      %total pages%
         *astrng* _ *astrng*,
            "Total pages in file = ", STRING(fstfdb[11B].RH), EOL;
      %used words%
         *astrng* _ *astrng*,
            STRING(usd), " words used out of ", STRING(tot),
            " words in file (=", STRING(percent _
            (usd*100+tot/2)/tot), "%)";
      %percent used too low?%
         IF percent < 70 AND fstfdb[11B].RH > 3 THEN
            *astrng* _ *astrng*, EOL, EOL,
               "Try an Update File Compact to improve % used";
      END;
   RETURN
   END.

%.....group allocation.....%
   %
   This code provides for "deleteing" jobs from the group allocation
   data page for the nls commands "Execute Logout"
   %
   (lockpage) PROCEDURE(core); %lock the data page%                    12B
      LOCAL
         count;
      FOR count _ 20 DOWN UNTIL = 0 DO
         BEGIN
         IF SKIP !AOSF @core THEN
            BEGIN
            [core+1] _ gtadcall();  %get time; set time of locking%
            RETURN(TRUE);
            END;
         !disms (1000); %wait a sec%
         END;
      RETURN(FALSE);
      END.
%.....CRLF => EOL.....%
```

```
(cnvcrlfteol) PROCEDURE(string); %convert string with CRLF's to
EOL's%                                                          13A
   LOCAL srcptr, desptr, hptr, cnt, i;
   REF string;
   srcptr _ desptr _ chbptr(0) + &string;
   cnt _ string.L;
   i _ 0;
   WHILE i < cnt DO
      BEGIN
      IF (^desptr _ ^srcptr) = CR THEN
         BEGIN
         hptr _ srcptr;
         IF ^srcptr = LF THEN
            BEGIN
            BUMP DOWN cnt;
            .desptr _ EOL;
            END
         ELSE srcptr _ hptr;
         END;
      BUMP i;
      END;
   string.L _ cnt;
   ^desptr _ 0;
   RETURN;
   END.
(cnveoltcrlf)PROCEDURE(t1, string); %convert string with EOL's to
CRLF's%                                                         13B
   %start at pointer t1, go to string end.  put converted stuff in
   string%
   LOCAL TEXTPOINTER t4;
   REF t1, string;
   *string* _ NULL;
   FIND SE(t1) ^t4 > ;
   cetcapp(&t1, $t4, &string);
   RETURN;
   END.
(cetcapp)PROCEDURE(t1, t4, string); %eol-to-crlf convert string
segment t1 t4, append to string%                                13C
   %start at pointer t1, go to string end.  put converted stuff in
   string%
   LOCAL TEXT POINTER t2, t3;
   REF t1, string, t4;
   WHILE (FIND BETWEEN t1 t4 (^t2 [EOL] ^t1 ^t3 _ t3)) DO *string* _
   *string*, t2 t3, CR, LF;
   *string* _ *string*, t1 t4;
   RETURN;
   END.
FINISH of AUXCOD



%....Moved to here by KJM 7-JUN-77  ....%
   (nmdlbset) PROCEDURE  % set name delimiters for a branch %    15A
      (stid1, dlleft, dlrght, da);                              15A1
      xnmdlset (stid1, stid1, dlleft, dlrght, da);
```

```
    RETURN;
    END.

(nmdlpset) PROCEDURE  % set name delimiters for a plex %          15B
    (stid1, dlleft, dlrght, da);                                  15B1
    LOCAL stid2;
    stid1 _ plxset (stid1 :stid2);
    xnmdlset (stid1, stid2, dlleft, dlrght, da);
    RETURN;
    END.

%...call stack underflow routine...%
(uflow) PROCEDURE; %general stack underflow routine%              16A
    S _ M _ -$gstksz;
    !HRL M,M; !HRL S,S;
    !HRRI S,gstack; !HRRI M,gstack;
    state _ $gstack + 2;
    state[1] _ $supervisor;
    state[2] _ $gstack;
    state[3] _ $edit;
    dismes(2, $"Call stack underflow -- report circumstances to ARC
    staff");
    supervisor();
    halt();
    END.

%...antique typeout routines...%
(typeas) PROCEDURE (astrng); %Type a-string on tty%               17A
    %-----------------%
    REF astrng;
    IF NOT astrng.L THEN RETURN;
    !sout(101B, chbmty + $astrng, -astrng.L);
    RETURN;
    END.

(typech) PROCEDURE(char); %type (translated) character on tty%    17B
    LOCAL tchar, spclcharstr;
    REF spclcharstr;
    %-----------------%
    %
    IF (R1 _ trnslo[char]) NOT= nullch THEN !pbout();
    %
    RETURN
    END.

(crlf) PROCEDURE; %type a carriage return-line feed%              17C
    %-----------------%
    %
    !pbout(trnslo[EOL]);
    %
    RETURN;
    END.

% give warning message if experimental system %
(xwarning) PROC(curptr, parsemode);                               18A
    % display a warning message to the user if he is using the
```

```
    experimental system and returns TRUE %
    REF curptr;
    %-----------%
    CASE parsemode OF
       = parsing:
          IF jdebug THEN
             dismes (2, $"WARNING: EXPERIMENTAL SYSTEM, use at your
             own risk!");
       ENDCASE;
    RETURN (&curptr);
    END.


%......query support routines......%
  (qport) PROCEDURE(qflg); % Query language initialization. %        19A

    % This is the entry point into the query language. %
    % note that the flag qflg is not used!  Originally used for
    differentiating between "query" and "NIC Resource Query %
    % If global flag nlparse is FALSE we came from Exec directly and
    exit will be by execute quit; if nlparse is TRUE we came from nls
    and exit will do a jump file return.%

    LOCAL STRING com[100], filename[50];
    INVOKE (sigquery, RETURN);
    crlf();
    qustart($filename, $com);
    DROP (sigquery);
    RETURN;

    (sigquery) CATCHPHRASE;                                          19A11
       BEGIN
       DISABLE (sigquery);
       crlf();
       typeas($"Error exit from query");
       IF NOT nlparse THEN %ceq()% TERMINATE  %RETURN%
       ELSE
          BEGIN
          ququit();      %Jump file return%
          %GOTO STATE;%
          TERMINATE;  %RETURN%
          END;
       END;

    END.
  (qustart) PROCEDURE(filename, com);  % Query parser. %             19B
    % This parser accepts a Bring command and the name of one of four
    basic files. Interrogation of a file continues with the Show
    command. Quit causes exit back to EXEC or TNLS. A question mark
    will give command language description to user.%

    LOCAL wkstid, loaded, param1, resptr[10];
    LOCAL STRING apndir[50], tempsr[50];
    REF filename, com, param1;

    % Set up default directory string %
```

```
    !JSYS gjinf;
    gdname(R2,$tempsr);
    *tempsr* _ "<,*tempsr*,">;
*apndir* _ "<NETINFO>";
*filename* _ "<NETINFO>HELP1";
loaded _ FALSE;
crlf();
% load first "help" file %
    wkstid _ guloadit( &filename, $apndir);
INVOKE (sigparse, rptloop);
(rptloop):                                                    19B13
LOOP % parsing loop %
    BEGIN
    echon();
    crlf();
    typeas($"-");
    CASE inpcuc() OF
        ="S: % Show command %
            BEGIN
            echo($"how ");
            quinlit(&com);
            deblank(&com);
            IF loaded THEN % file is present and loaded %
                gl(wkstid, &com)
            ELSE
                BEGIN
                typeas($"You have not specified a file yet.");
                END;
            END;
        ="B: % Bring (file name) %
            BEGIN
            echo($"ring ");
            quinlit(&com);              % read filename %
            deblank(&com);
            *filename* _ *com*;
            loaded _ TRUE;
            wkstid _  guloadit( &filename, $apndir);
            END;
        ="D: % data base display %
            BEGIN
            echo($"ata Base of user files");
            CASE inpcuc() OF
                = CA, = EOL:
                    BEGIN
                    *filename* _  "<NETINFO>DATABASE";
                    loaded _ TRUE;
                    wkstid _  guloadit( &filename, $apndir);
                    END;
                ENDCASE REPEAT LOOP;
            END;
        ="R: % Resource notebook display %
            BEGIN
            echo($"esource Notebook");
            CASE inpcuc() OF
                = CA, = EOL:
                    BEGIN
```

```
            *filename* _  "<NETINFO>RESOURCES";
            *apndir* _ "<NETINFO>";
            loaded _ TRUE;
            wkstid _ guloadit( &filename, $apndir );
            END;
         ENDCASE REPEAT LOOP;
      END;
   ="A: % arpanet news display %
      BEGIN
      echo($"RPANET NEWS");
      CASE inpcuc() OF
         = CA, = EOL:
            BEGIN
            *filename* _  "<HELP>ARPANEWS";
            *apndir* _ "<NETINFO>";
            loaded _ TRUE;
            wkstid _ guloadit( &filename, $apndir );
            END;
         ENDCASE REPEAT LOOP;
      END;
   ="I: % Ident file display %
      BEGIN
      echo($"dent File");
      CASE inpcuc() OF
         = CA, = EOL:
            BEGIN
            *filename* _  "<IDENTFILE>IDENTS.MASTER";
            loaded _ TRUE;
            wkstid _ guloadit( &filename, $apndir );
            END;
         ENDCASE REPEAT LOOP;
      END;
   ="N: % Start over -- NIC command %
      BEGIN
      echo($"ic");
      CASE inpcuc() OF
         = CA, = EOL:
            BEGIN
            *filename* _  "<NETINFO>HELP1";
            loaded _ FALSE;
            wkstid _  guloadit( &filename, $apndir);
            END;
         ENDCASE REPEAT LOOP;
      END;
   ="?: % help, then back to previous file if any %
      BEGIN
      CASE inpcuc() OF
         = CA, = EOL:
            BEGIN
            crlf();
            wkstid _ guloadit( $"<NETINFO>HELP", $apndir );
            IF loaded THEN
               BEGIN
               typeas($" Please wait ");
               wkstid _ guloadit( &filename, $apndir );
               END;
```

```
            END;
         ENDCASE REPEAT LOOP;
      END;
   ="H: % help, then back to previous file if any %
      BEGIN
      echo($"elp");
      CASE inpcuc() OF
         = CA, = EOL:
            BEGIN
            crlf();
            wkstid _ guloadit( $"<NETINFO>HELP", $apndir );
            IF loaded THEN
               BEGIN
               typeas($" Please wait ");
               wkstid _ guloadit( &filename, $apndir );
               END;
            END;
         ENDCASE REPEAT LOOP;
      END;
   %="V:  viewspecs  not needed (show and bring both force
   viewspecs)%
      %BEGIN%
      %echo($"iewspecs: Type ");%
      %getvsp(); gets string from keyboard%
      %treslev(tda.dacsp); take care of relative levels%
      %&param1 _ xviewspecs($resptr,1);
      cspupdate _ lda();
      cspvs _ param1;
      cspvs[1] _ param1[1] ;
      dpset(dspyes,[cspupdate].dacsp,erdfil,endfil);
      cmdfinish();
      END;%
   ="Q: % quit %
      BEGIN
      echo($"uit ");
      CASE inpcuc() OF
         = CA, = EOL:
            BEGIN
            IF NOT nlparse THEN %ceq()% RETURN
            ELSE
               BEGIN
               %Jump file return%
               guquit();
               %GOTO oldgps;%
               RETURN;
               END;
            END;
         ENDCASE REPEAT LOOP;
      END;
   ENDCASE
      BEGIN
      crlf();
      typeas($"Not recognized");
      END;
END;
```

```
   (sigparse) CATCHPHRASE;                                          19B16
      BEGIN
      DISABLE (sigparse);
      CASE SIGNAL OF
         = ofilerr:
            BEGIN
            IF NOT (FIND SF(*filename*) ["<]) THEN % no directory
            name %
               BEGIN
               *apndir* _ *tempsr*;  % append user directory %
               INVOKE (sigreal, realerr);
               wkstid _ quloadit( &filename, $apndir); % Try with
               different directory name %
               END;
            (realerr):                                          19B16C1C
            IF sysmsg THEN
               BEGIN
               typeas(MESSAGE); % there WAS a dir, or failed twice %
               crlf();
               typeas($"File not found");
               sysmsg _ 0;
               END;
            *apndir* _ "<NETINFO>"; % restore default directory %
            DROP (sigreal);
            TERMINATE;        %GOTO rptloop%
            END;
         =statesig: TERMINATE;        %GOTO rptloop%
         ENDCASE
            CONTINUE;
      END;
   (sigreal) CATCHPHRASE;                                           19B17
      BEGIN
      DISABLE (sigreal);
      TERMINATE;      %GOTO realerr;%
      END;

   END.

(quloadit) PROCEDURE( filename, apndir );                           19C
   LOCAL fileno, wkstid;
   LOCAL STRING qsname[100];
   REF filename, apndir;
   *qsname* _ *filename*;
   IF NOT (FIND SF(*qsname*) ["<]) THEN % filename has no directory
   %
      *qsname* _ *apndir*, *filename*;
   %xlf($qsname, &tda);
   sysmsg _ 0;
   freflnt();
   fileno _ tda.dacsp.stfile;%
   % --- new code ---%
      fileno _ cloafil($qsname);
      curmkr _ orgstid;
      curmkr.stfile _ fileno;
      curmkr[1] _ 1;
   *apndir* _ "<NETINFO>";
```

```
    crlf();
    typeas($"--------");
    feedit(&tda,$"Bw");
    wkstid _ origin;
    wkstid.stfile _ fileno;
    IF (wkstid _ getsub(wkstid)).stpsid = origin THEN
       BEGIN
       typeas($"File is empty");
       END
    ELSE
       BEGIN
       feedit(&tda, $"esb");
       printg(&tda,wkstid, wkstid, brnchv,0);
       END;
    crlf();
    typeas($"--------");
    RETURN( wkstid );
    END.

(q1) PROCEDURE (orstid, com); %converts "show" to appropriate jump%
                                                                    19D

    % given the string typed by the user, find a statement by that
    name in the current file.%

    LOCAL prvstid, wkstid;
    LOCAL TEXT POINTER z1, z2, z3, z4;
    LOCAL STRING com1[100], erout[100];
    REF com;
    %Initialize stids %
       wkstid _ prvstid _ orstid;
    feedit(&tda, $"Bw"); % Don't indent; all lines, all levels %
    *erout* _ NULL;
    IF NOT (FIND SF(*com*) ^z1 [':] ^z2 _z2 ^z3 [ENDCHR] ^z4) THEN

       % User has typed a command like: show xyz CR %

       BEGIN
       wkstid _ namingrp(prvstid, prvstid, &com, 1000);
       IF wkstid # endfil THEN
          BEGIN
          quprout( wkstid );
          RETURN;
          END
       ELSE
          BEGIN
          wkstid _ namingrp(orstid, endfil, &com, 1000);
          IF wkstid # endfil THEN
             BEGIN
             quprout( wkstid );
             RETURN;
             END;
          crlf();
          *erout* _ *com*," not found.";
          typeas($erout);
          RETURN;
```

```
      END;
   END
ELSE
   % User has typed: show xyz:abc CR
   which means: Find a branch named abc anywhere within the
   branch named xyz. %

   BEGIN
   *com1* _ z3 z4;
   *com* _ z1 z2;
   wkstid _ namingrp(orstid, endfil, &com, 1000);
   IF wkstid = endfil THEN
      BEGIN
      crlf();
      *erout* _ *com*," not found.";
      typeas($erout);
      RETURN;
      END;
   prvstid _ wkstid;
   wkstid _ namingrp(prvstid, prvstid, $com1,1000);
   IF wkstid # endfil THEN
      BEGIN
      guprout( wkstid );
      RETURN;
      END
   ELSE
      BEGIN
      crlf();
      *erout* _ *com1*, " not found under ", *com*, ".;
      typeas($erout);
      RETURN;
      END;
   END;
END.

(guprout) PROCEDURE (stid);                                    19E
   feedit(&tda, $"esb");
   gubing(stid);
   printg(&tda,stid, stid, brnchv,0);
   RETURN;
   END.

(gubing) PROCEDURE (stid); % process embedded viewspecs.%      19F

   % When a statement name is followed by a string such as (uv:)
   this procedure recognizes uv as viewspecs and turns them on.%

   LOCAL STRING vstrng[10];
   LOCAL char;
   *vstrng* _ NULL;
   s2work _ stid;
   s2work[1] _ fchtxt(getsdb(stid));
   fechc1(forward,$s2work);
   IF (char _ READC($s2work)) # "( THEN RETURN;
   IF (char _ READC($s2work)) # ": THEN RETURN;
   LOOP
```

```
      BEGIN
      char _ READC($s2work);
      IF char # ") THEN *vstrng* _ *vstrng*,char
      ELSE
         BEGIN
         feedIt(&tda, $vstrng);
         RETURN;
         END;
      END;
   END.

(guquit) PROCEDURE; % Quit and restore nls context. %                    19C

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

   LOCAL STRING str[100];
   %gadjt($b1, $popls, $str, 1, &tda);%
   %mvcbl();%
   %dismes(2,$"Ququit Called");%
   RETURN;
   END.

(quinlit) PROCEDURE(com); %read a string, handle special
characters.%                                                            19H

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

   LOCAL char;
   REF com;
   LOOP
      BEGIN
      *com* _ NULL;
      LOOP
         BEGIN
         char _ inpcuc();
         CASE char OF
            =BC: IF com.L > empty THEN
               BEGIN
               todco(*com*[com.L]);
               bkc(&com);
               END;
            =CD: SIGNAL(statesig, 0);
            =CA: RETURN(TRUE);
            =EOL: RETURN(TRUE);
            ="?:
               BEGIN
               *com* _"H ;
               RETURN(TRUE);
               END;
            ENDCASE *com* _ *com*, char;
         END;
```

```
      END;
   END.

(deblank) PROCEDURE (string);  %deblank the string%                    19I

   % Eliminate leading blanks for show and bring command. %

   LOCAL TEXT POINTER z1, z2;
   REF string;
   IF FIND SF(*string*) ^z1 $NP ^z2 THEN
      ST z1 z2 _ NULL;
   RETURN;
   END.
%... Moved to here by ROM ...%
(brkconnection)  PROCEDURE; %break display screen connection%          20A
   %set program counter to actual routine%
   !MOVEM R1,svac1;
   !MOVEI R1,brkclabel;
   !MOVEM R1,@levtab;
   !MOVE R1,svac1;
   !JSYS debrk;
   (brkclabel):                                                        20A7
      rstconnection();
      GOTO STATE;
   END.

(rstconnection)  PROCEDURE; %break display screen connection%          20B
   %break links%
      IF NOT SKIP !tlink(6B11 .V 777777B, 777777B) THEN NULL;
      %break adviz%
      IF NOT SKIP !adviz(4B11) THEN NULL;
   %shut down NLS display%
      shutdis();
   IF ldspjfn THEN
      BEGIN
      IF NOT SKIP !closf(ldspjfn) THEN NULL;
      reljfn(ldspjfn);
      END;
   linkcnsl _ ldspjfn _ 0;
   %restore NLS display%
      %restore old format%
         IF savnldevice NOT= nldevice THEN setdev(savnldevice);
      continue _ TRUE;
      <INTNLS, initdis>();
      continue _ FALSE;
      alldsp();
   chntab[1] _ savchntab[1];
   RETURN;
   END.
```

```
< NINE, BACKVIEW.NLS;7, >, 30-May-78 16:20 HGL ;;;;
FILE backview % <ARCSUBSYS>L109  <RELNINE>BACKVIEW % %
(ARCSUBSYS,L109,) (RELNINE,backview.rel,) %
ALLOW!


%Viewspec acceptors%

   (stklt) PROCEDURE(setchr, vs1, vs2);                                    5B
      %Puts input viewspec characters on stack, appends them to string
      vspstr, and calls setlt to activate them.%
      %--------------%
      LOCAL vspec[2];
      vspec _ vs1;
      vspec[1] _ vs2;
      IF setchr = BC THEN
         BEGIN
         bkc($vspstr);
         POP savevspec TO vspec;
         RETURN(vspec, vspec[1]);
         END
      ELSE PUSH vspec ON savevspec;
      vspec _ setlt(setchr, vs1, vs2:vspec[1]);
      *vspstr* _ *vspstr*, setchr;
      RETURN(vspec, vspec[1]);
      END.

   (feedlt)  PROCEDURE(dpa, astrng);                                       5C
      %Given the address of a display area, this routine changes
      its vspecs in accord with the specificaations in the
      A-string passed it.  It passes the characters in the
      A-string to <PRMSPC, SETLT>, except for content analyzer
      patterns.%
      %------------%
      LOCAL count, length, char, vs1, vs2;
      LOCAL TEXT POINTER tp;
      LOCAL STRING castrg[250];
      REF dpa, astrng;
      vs1 _ dpa.davspec;
      vs2 _ dpa.davspc2;
      FOR count _ 1 UP UNTIL > astrng.L DO
         IF (char _ *astrng*[count]) = "; THEN
            BEGIN
            *castng* _ NULL;
            UNTIL (char _ *astrng*[count _ count + 1]) = "; DO
               BEGIN
               *castng* _ *castng*, char;
               IF count >= length THEN EXIT LOOP1;
               END;
            FIND SF(*castng*) ^tp;
            IF ccompile(
               53       % content%,
               tp,      % starting stid %
               0,       % default compiler %
               0,       % default output file (none) %
               -1,      % no messages %
```

```
            0,        % no viewspecs -- include all %
            &dpa      % display area % )
          THEN % error in compiling or loading %
            err($"illegal pattern  -- not instituted");
          END
      ' ELSE vs1 _ setlt(char, vs1, vs2 : vs2);
    dpa.davspec _ vs1;
    dpa.davspc2 _ vs2;
    RETURN;
    END.


(setlt)                                                          5E
    %This routine adjusts the viewspecs in accord with characters
    entered during view specification.  Saves the viewspec words on
    the stack savevspecfor each character input.  when a BC is input
    the stack is poped.  The strring being displayed in the name area
    is updated accordingly.%
    %---------------%
    PROCEDURE(setchr, vs1, vs2);
    LOCAL goodvs, settmp, vspec[2];
    vspec _ vs1;
    vspec[1] _ vs2;
    goodvs _ TRUE;
    CASE setchr OF
       ='a: % l_l-1 %
          IF vspec.vsrlev THEN
             BEGIN
             IF vspec.vslev = 0 OR vspec.vslevd THEN
                BEGIN
                BUMP vspec.vslev;
                vspec.vslevd _ TRUE;
                END
             ELSE BUMP DOWN vspec.vslev;
             END
          ELSE IF vspec.vslev > 0 THEN BUMP DOWN vspec.vslev;
       ='b:  % l_l+1 %
          IF vspec.vslevd THEN
             BEGIN
             BUMP DOWN vspec.vslev;
             IF vspec.vslev = 0 THEN vspec.vslevd _ FALSE;
             END
          ELSE IF vspec.vslev < 63 THEN BUMP vspec.vslev;
       ='c:  % l_all %
          BEGIN
          vspec.vslev _ 63;
          vspec.vsrlev _ vspec.vslevd _ FALSE;
          END;
       ='d:  % l_1 %
          BEGIN
          vspec.vslev _ 1;
          vspec.vsrlev _ vspec.vslevd _ FALSE;
          END;
       ='e:  % l=rel %
          BEGIN
          IF vspec.vsrlev THEN %user is already in e state, reset
```

```
        previous e%
            BEGIN
            (rstlev):                                          5E8E2B
            vspec.vslev _ vspec.vsrlev;
            vspec.vsrlev _ FALSE;
            vspec.vslevd _ FALSE;
            END;
        vspec.vsrlev _ IF vspec.vslev THEN vspec.vslev ELSE TRUE;
        vspec.vslev _ FALSE;
        vspec.vslevd _ FALSE;
        END;
    ="g:  % branch only on %
        BEGIN
        vspec.vsbrof _ TRUE;
        vspec.vsplxf _ FALSE;
        END;
    ="h:  % branch only / plex only off %
        BEGIN
        vspec.vsbrof _ FALSE;
        vspec.vsplxf _ FALSE;
        END;
    ="i:  % content analyzer success %
        BEGIN
        vspec.vscapf _ TRUE;
        vspec.vscakf _ FALSE;
        END;
    ="j:  % content analyzer off %
        BEGIN
        vspec.vscapf _ FALSE;
        vspec.vscakf _ FALSE;
        END;
    ="k:  % content analyzer k flag %
        % only use content analyzer for first statement in sequence
        %
        BEGIN
        vspec.vscakf _ TRUE;
        vspec.vscapf _ FALSE;
        END;
    ="l:  % plex only on %
        BEGIN
        vspec.vsplxf _ TRUE;
        vspec.vsbrof _ FALSE;
        END;
    ="m:  vspec.vsstnf _ TRUE;  % location numbers on %
    ="n:  vspec .vsstnf _ FALSE;  % location numbers off %
    ="o:  vspec.vsfrzf _ TRUE;  % frozen on %
    ="p:  vspec.vsfrzf _ FALSE;  % frozen off %
    ="q:  % t_t-1 %
        IF vspec.vstrnc > 1 THEN BUMP DOWN vspec.vstrnc;
            %if this is changed so 0 is allowed, dafrmt must be
            fixed so it doesn't try to freeze every page in the
            file%
    ="r:  % t_t+1 %
        IF vspec.vstrnc < 63 THEN BUMP vspec.vstrnc;
    ="s:  % t_all %
        vspec.vstrnc _ 63;
```

```
        ="t:  % t_1 %
           vspec.vstrnc _ 1;
        ="u:  % display area formatter on %
           vspec.vsdaft _ TRUE;
        ="v:  % display area formatter off %
           vspec.vsdaft _ FALSE;
        ="w:  % l=t=all %
           BEGIN
           vspec.vstrnc _ 63;
           vspec.vslev _ 63;
           vspec.vsrlev _ vspec.vslevd _ FALSE;
           END;
        ="x:  % l=t=1 %
           BEGIN
           vspec.vstrnc _ vspec.vslev _ 1;
           vspec.vsrlev _ vspec.vslevd _ FALSE;
           END;
        ="y:  vspec.vsblkf _ TRUE;  % blank line on %
        ="z:  vspec.vsblkf _ FALSE;  % blank line off %
        ="A:  % indenting on, relative indenting off %
           BEGIN
           vspec.vsindf _ TRUE;
           vspec.vsrind _ FALSE;
           END;
        ="B:  % indenting off, relative indenting off %
           BEGIN
           vspec.vsindf _ FALSE;
           vspec.vsrind _ FALSE;
           END;
        ="C:  vspec.vsnamf _ TRUE;   % names on %
        ="D:  vspec.vsnamf _ FALSE;  % names off %
        ="E:  vspec.vspagf _ TRUE;   % Paging on %
        ="F:  vspec.vspagf _ FALSE;  % Paging off %
        ="G:  vspec.vsstnr _ TRUE;   % statement numbers on right %
        ="H:  vspec.vsstnr _ FALSE;  % statement numbers on left %
        ="I:  vspec.vssidf _ TRUE;   % sid flag on %
        ="J:  vspec.vssidf _ FALSE;   % sid flag off %
        ="K:  vspec.vsidtf _ TRUE;   % initials, date, on %
        ="L:  vspec.vsidtf _ FALSE;  % initials, date, off %
        ="O:  vspec.vsusqf _ TRUE;   % user sequence generator on %
        ="P:  vspec.vsusqf _ FALSE;  % user sequence generator off %
        ="Q:  % indenting and relative indenting on %
           BEGIN
           vspec.vsrind _ TRUE;
           vspec.vsindf _ TRUE;  % so Output Processors will be sure
           to show indenting on %
           END;
        ENDCASE goodvs _ FALSE;
     RETURN(vspec, vspec[1], goodvs);
     END.


% CLIST UTILITY %
   % modularized clist procedure %
      (clist)   PROCEDURE (type, fno1, fno2);  %build clist%          6A1
         %This routine constructs the correspondence list according to
```

```
    the parameters passed as follows: See clhdr and clistr for
    format of clist.
       If either fno1 or fno2 are > 0 then only stid's
       belonging to those files are used.
       if type = 0 then clhead.clcnt is set to zero.
       if type .A 1 = 1 then the csp's are used.
       if type .A 2 = 2 then the frozen list entries are
       used.
       if type .A 4 = 4 then the return ring entries are used.
       if type .A 8 = 8 then the markers are used.
       any combination of the above is valid.%
    %----------------%
    LOCAL end1, entry1, cl, cb;
    LOCAL STRING fnam1[150], fnam2[150];
    cl _ clinit(type, fno1, fno2, $fnam1, $fnam2);
    IF type = 0 THEN RETURN;
    end1 _ (entry1 _ $dpyarea) + dacnt*dal;
    UNTIL entry1 >= end1 DO
       BEGIN
       cl _ clbody(entry1, type, cl, fno1, fno2, $fnam1, $fnam2);
       entry1 _ entry1 + dal;
       END;
    RETURN;
    END.

(clinit)  PROCEDURE (type, fno1, fno2, fnam1, fnam2);          6A2
    %This procedure does the initialization for clist.  Pulled out
    to allow calling clist for only one display area.%
    %----------------%
    LOCAL cl;
    REF fnam1, fnam2;
    clhead.clbuff _ cl _ clistaddr;
    clhead.clcnt _ 0;
    clhead.clfno1 _ fno1;
    clhead.clfno2 _ fno2;
    clhead.cltype _ type;
    IF type .A 4 THEN %file return ring%
       BEGIN
       %do not trust file numbers--use file name%
       *fnam1* _ NULL;
       *fnam2* _ NULL;
       IF fno1 THEN filnam(fno1, &fnam1);
       IF fno2 THEN filnam(fno2, &fnam2);
       END;
    RETURN(cl);
    END.
(clbody)  PROCEDURE (da, type, cl REF, fno1, fno2, fnam1, fnam2);
                                                               6A3
    %This procedure does the body of the clist operation for only
    the display area specified.  It was pulled out so a clist can
    be done for only one da.  It returns the address of the next
    available record slot.%
    %----------------%
    LOCAL fz, hd, mk, last1, a, b, srr, frr, i, j, fn;
    REF da, fz, mk, srr, frr, fn, fnam1, fnam2;
    IF type .A 1 THEN %csp's%
```

```
        BEGIN
        IF da.daexis AND NOT da.daempty THEN
           IF (fno1 = 0 AND fno2 = 0) OR
            (da.dacsp.stfile = fno1) OR
            (da.dacsp.stfile = fno2) THEN
              BEGIN
              cl.clst1 _ da.dacsp;
              cl.clcc1 _ da.daccnt;
              cl.clst2 _ endfil;
              cl.clcc2 _ 1;
              cl.clfixed _ FALSE;
              &cl _ upclptr(&cl);  %get address for next entry%
              BUMP clhead.clcnt;
              END
           ELSE NULL;
        END;
     IF type .A 2 THEN %frozen list%
        BEGIN
        IF da.daexis AND NOT da.daempty AND &fz _
        da.dafrz1 THEN
           DO
              IF (fno1 = 0 AND fno2 = 0) OR
               (fz.fzstid.stfile = fno1) OR
               (fz.fzstid.stfile = fno2) THEN
                 BEGIN
                 cl.clst1 _ fz.fzstid;
                 cl.clcc1 _ cl.clcc2 _ 1;
                 cl.clst2 _ endfil;
                 cl.clfixed _ TRUE;
                 &cl _ upclptr(&cl);  %get address for next entry%
                 BUMP clhead.clcnt;
                 END
              ELSE NULL
           UNTIL (&fz _ fz.fznext) = 0;
        END;
     IF type .A 4 THEN %file return ring%
        BEGIN
        IF da.daexis THEN
           BEGIN
           IF &frr _ da.dalink THEN
              BEGIN
              INVOKE (repeatloop, rptlloop); %in case top entry
              non-existant%
              DISABLE (repeatloop);
              FOR i _ frrlength(&frr) DOWN UNTIL < 0 DO
                 BEGIN
                 ENABLE (repeatloop);
                 &fn _ readfrring(&frr, i : &srr);
                 DISABLE (repeatloop);
                 IF (fno1 = 0 AND fno2 = 0)
                 OR (fno1 AND a _ (*fn* = *fnam1*))
                 OR (fno2 AND b _ (*fn* = *fnam2*)) THEN
                    BEGIN
                    FOR j _ srrlength(&srr) DOWN UNTIL < 0 DO
                       BEGIN
                       cl.clst1 _ readsrring(&srr, j : cl.clcc1);
```

```
                        cl.clst1.stfile _
                           IF a THEN fno1 ELSE fno2;
                        cl.clst2 _ endfil;
                        cl.clcc2 _ 1;
                        cl.clfixed _ FALSE;
                        &cl _ upclptr(&cl);  %addr. for next entry%
                        BUMP clhead.clcnt;
                        END;
                     END;
                  (rpt1loop):                                    6A3G2B4F
                  END;
               DROP (repeatloop);
               END
            ELSE err($"no file return ring in clist");
            END;
         END;
      IF type .A 8 THEN %markers%
         BEGIN
         IF da.daexis AND NOT da.daempty THEN
            IF  (fno1 = 0 AND fno2 = 0) OR
            (da.dacsp.stfile = fno1) OR
            (da.dacsp.stfile = fno2) THEN
               BEGIN
               &mk _ $mkrtb - $filhed + (hd _
                  filhdr(da.dacsp.stfile));
               last1 _ &mk + [$mkrtbl - $filhed + hdl * mkrl;
               %bounds check%
                  IF (last1 - &mk) > [$mkrtxn + hd - $filhedl THEN
                     BEGIN
                     [$mkrtbl - $filhed + hdl _ [$mkrtxn + hd -
                     $filhedl / mkrl;
                     last1 _ &mk + [$mkrtbl - $filhed + hdl * mkrl;
                     END;
               UNTIL &mk >= last1 DO
                  BEGIN
                  cl.clst1 _ 0;
                  cl.clst1.stfile _ da.dacsp.stfile;
                  cl.clst1.stpsid _ mk.mkpsid;
                  cl.clcc1 _ mk.mkccnt;
                  cl.clst2 _ endfil;
                  cl.clcc2 _ 1;
                  cl.clfixed _ FALSE;
                  &mk _ &mk + mkrl;
                  &cl _ upclptr(&cl);  %get address for next entry%
                  BUMP clhead.clcnt;
                  END;
               END
            ELSE NULL;
         END;
      RETURN(&cl);
      (repeatloop) CATCHPHRASE();                                  6A3J
         BEGIN
         CASE SIGNALTYPE OF
            = notetype :  NULL;
            = helptype :  NULL;
            = aborttype :
```

```
            IF SIGNAL = frremptyentry THEN TERMINATE;
         ENDCASE;
      DISABLE (repeatloop);
      CONTINUE;
      END;
   END.
(upclptr) PROCEDURE (claddr);  %get next tree record slot%      6A4
   LOCAL cb;
   claddr _ claddr + cll;  %increment to point to next record
   slot%
   IF claddr > clistaddr + clsize - cll THEN
      BEGIN  %out of space%
      %allocate a bigger block for the correspondence list%
         cb _ getblk(clsize+100, $dspblk);
      mvbfbf(clistaddr, cb, clsize);   %move old block to new%
      claddr _ cb + (claddr - clistaddr);  %point to new block%
      freeblk(clistaddr, $dspblk);  %free old buffer%
      clistaddr _ clhead.clbuff _ cb;  %point to new block%
      clsize _ clsize + 100;          %update size%
      %set return value to next record slot in new block%
         claddr _ clistaddr + (clhead.clcnt +1) * cll;
      END;
   RETURN(claddr);
   END.

% modularized clupdt procedure %
   (clupdt)  PROCEDURE;  %unbuild clist%                          6B1
      %This routine updates various things from the correspondence
      list according to the parameters in the clist header (clhead):
         If either clhead.clfno1 or clhead.clfno2 are > 0 then
         only stid's belonging to those files are updated.
         if cltype = 0 then none of the following is changed.
         if cltype .A 1 = 1 then the csp's are updated.
         if cltype .A 2 = 2 then the frozen lists are updated.
         if cltype .A 4 = 4 then the return ring is updated.
         if cltype .A 8 = 8 then the markers are updated.
         any combination of the above is valid.
      See clhdr and clistr for format of clist.%
         % if clst1 = endfil, then this statement has been deleted
         from the file and clst2 points to the "next" statement in
         the file. if stcc1 changes then the text that used to be
         pointed to has been deleted or moved and clcc1 has been
         updated to something reasonable.%
      %----------------%
      LOCAL end1, entry1, cl;
      LOCAL STRING fnam1[150], fnam2[150];
      REF entry1;
      IF clhead.cltype = 0 THEN RETURN;
      cl _ cluinit(clhead.cltype, $fnam1, $fnam2);
      end1 _ (&entry1 _ $dpyarea) + dacnt*dal;
      UNTIL &entry1 = end1 DO
         BEGIN
         cl _ clubody(&entry1, clhead.cltype, cl, $fnam1, $fnam2);
         &entry1 _ &entry1 + dal;
         END;
      RETURN;
```

```
    END.

(cluinit) PROCEDURE (type, fnam1, fnam2);                          682
    %This procedure does the initialization for clupdt.  Pulled
    out so clupdt can be performed for only one display area.
    Returns address of beginning of correspondence list%
    %----------------%
    LOCAL cl;
    REF fnam1, fnam2;
    cl _ clhead.clbuff;
    IF type .A 4 THEN %file return ring%
       BEGIN
       %do not trust file numbers--use file name%
       *fnam1* _ NULL;
       *fnam2* _ NULL;
       IF clhead.clfno1 THEN filnam(clhead.clfno1, &fnam1);
       IF clhead.clfno2 THEN filnam(clhead.clfno2, &fnam2);
       END;
    RETURN(cl);
    END.

(clubody) PROCEDURE (da, type, cl REF, fnam1, fnam2);              683
    %This procedure does the updating for one display area for
    clupdt.  Pulled out so clupdt can be performed for only one
    display area.  Returns address of next cl record.%
    %----------------%
    LOCAL stid, fz, fz2, hd, view1, view2, mklngth, mkold, mk,
    last1, srr, frr, fn, i, j, cc, a, b, curlength;
    LOCAL TEXT POINTER b1;
    REF da, fz, fz2, mklngth, mk, srr, frr, fn, fnam1, fnam2;
    IF clhead.cltype .A 1 THEN %csp's%
       BEGIN
       IF da.daexis AND NOT da.daempty THEN
          IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
          (clhead.clfno1 AND da.dacsp.stfile = clhead.clfno1) OR
          (clhead.clfno2 AND da.dacsp.stfile = clhead.clfno2) THEN
             BEGIN
             IF cl.clst1 = endfil OR cl.clst1.stfile NOT=
             da.dacsp.stfile THEN %replace by clst2%
                BEGIN
                IF cl.clst2 = endfil THEN
                   BEGIN % window is empty %
                   da.dacsp _ endfil;
                   da.daccnt _ 1;
                   da.daempty _ TRUE;
                   END
                ELSE
                   BEGIN % set current display to new file %
                   b1 _ cl.clst2;
                   b1[1] _ cl.clcc2;
                   % check if char cnt beyond end of statement %
                      IF b1[1] >= (curlength _ getstsize(b1)) THEN
                         b1[1] _ curlength;
                   da.dacsp _ b1;
                   da.daccnt _ b1[1];
                   END;
```

```
                END
             ELSE
                BEGIN
                IF da.dacsp NOT= cl.clst1 THEN
                   da.dacsp _ cl.clst1;
                IF da.daccnt NOT= cl.clcc1 THEN
                   da.daccnt _ cl.clcc1;
                END;
             &cl _ &cl + cll;
             END;
       END;
    IF clhead.cltype .A 2 THEN %frozen list%
       BEGIN
       IF da.daexis AND NOT da.daempty AND &fz _
       da.dafrz1 THEN
          DO
             IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
             (clhead.clfno1 AND fz.fzstid.stfile = clhead.clfno1)
             OR
             (clhead.clfno2 AND fz.fzstid.stfile = clhead.clfno2)
             THEN
                BEGIN
                IF cl.clst1 = endfil THEN
                   BEGIN %statement was deleted%
                   IF (&fz2 _ da.dafrz1) = &fz THEN
                      da.dafrz1 _ fz.fznext
                   ELSE
                      DO IF fz2.fznext = &fz THEN
                         BEGIN
                         fz2.fznext _ fz.fznext;
                         EXIT;
                         END
                      UNTIL (&fz2 _ fz2.fznext) = 0;
                   &fz2 _ fz.fznext;
                   fz.fznext _ fzfree := &fz;
                   IF &fz2 = 0 THEN EXIT LOOP;
                   END
                ELSE
                   IF cl.clst1 # fz.fzstid THEN
                      fz.fzstid _ cl.clst1;
                &cl _ &cl + cll;
                END
          UNTIL (&fz _ fz.fznext) = 0;
       END;
    IF clhead.cltype .A 4 THEN %file return ring%
       BEGIN
       IF da.daexis THEN
          BEGIN %cycle through frr and srr's%
          IF &frr _ da.dalink THEN
             BEGIN
             INVOKE (repeatloop, rpt2loop); %in case top entry
             non-existant%
             DISABLE (repeatloop);;
             FOR i _ frrlength(&frr) DOWN UNTIL < 0 DO
                BEGIN
                ENABLE (repeatloop);;
```

```
                &fn _ readfrring(&frr, i : &srr);
                DISABLE (repeatloop);;
                IF (clhead.clfno1 = 0 AND clhead.clfno2 = 0)
                OR (clhead.clfno1 AND a _ (*fn* = *fnam1*))
                OR (clhead.clfno2 AND b _ (*fn* = *fnam2*)) THEN
                   BEGIN %found a match -- update stid, cc for
                   each entry in srr%
                   FOR j _ srrlength(&srr) DOWN UNTIL < 0 DO
                      BEGIN
                      stid _ readsrring(&srr, j : cc, view1,
                      view2);
                      IF cl.clst1 NOT= endfil THEN
                         BEGIN
                         IF stid.stpsid NOT= cl.clst1.stpsid OR cc
                         NOT= cl.clcc1 THEN
                            storesrring(&srr, j, cl.clst1,
                            cl.clcc1, view1, view2)
                         END
                      ELSE
                         storesrring(&srr, j, cl.clst2, cl.clcc2,
                         view1, view2);
                      &cl _ &cl + cll;
                      END;
                   END;
                (rpt2loop):                                    6B3H2B4F
                END;
             DROP (repeatloop);
             END
          ELSE err($"no file return ring in clupdt");
          END;
       END;
    IF clhead.cltype .A 8 THEN %markers%
       BEGIN
       IF da.daexis AND NOT da.daempty THEN
          IF (clhead.clfno1= 0 AND clhead.clfno2= 0) OR
          (clhead.clfno1 AND da.dacsp.stfile= clhead.clfno1) OR
          (clhead.clfno2 AND da.dacsp.stfile= clhead.clfno2) THEN
             BEGIN
             &mk _ mkold _ $mkrtb - $filhed + (hd _
                filhdr(clhead.clfno1));
             last1 _ &mk + [&mklngth _ $mkrtbl - $filhed
             + hdJ*mkrl;
             UNTIL &mk = last1 DO
                BEGIN
                IF cl.clst1 = endfil THEN %delete marker%
                   BEGIN
                   mvbfbf(&mk + mkrl, &mk,
                   last1-&mk-mkrl);
                   BUMP DOWN mklngth;
                   last1 _ last1 - mkrl;
                   &cl _ &cl + cll;
                   END
                ELSE
                   BEGIN
                   IF mk.mkpsid NOT= cl.clst1.stpsid THEN
                      mk.mkpsid _ cl.clst1.stpsid;
```

```
                    IF mk.mkccnt NOT= cl.clcc1 THEN
                       mk.mkccnt _ cl.clcc1;
                    &cl _ &cl + cll;
                    &mk _ &mk + mkrl;
                    END;
                 END;
              END;
        END;
     RETURN(&cl);
     (repeatloop) CATCHPHRASE();                                    6B3K
        BEGIN
        CASE SIGNALTYPE OF
           = notetype :  NULL;
           = helptype :  NULL;
           = aborttype :
              IF SIGNAL = frremptyentry THEN TERMINATE;
           ENDCASE;
        DISABLE (repeatloop);
        CONTINUE;
        END;
     END.

(dpset) PROCEDURE (option, stid1, stid2, stopstid);                 6C
   %set global variables for recreate display routines.  In some
   cases, stid1 and stid2 are passed merely to indicate file
   involvement, not because they will be reformatted.%
   cdtype _ option;
   IF stid1 = stid2 THEN stid2 _ endfil;
   (cdstd1, cdstd2) _ (stid1, stid2);
   IF option = dsprfmt THEN %save statements before edit%
      BEGIN
      IF stid1 NOT= endfil THEN
         *cdstr1* _ $SF(stid1) SE(stid1);
      IF stid2 NOT= endfil THEN
         *cdstr2* _ $SF(stid2) SE(stid2);
      END;
   IF stopstid.stpsid = orgstid THEN cdstop _ endfil
   ELSE cdstop _ stopstid;
   RETURN
   END.

(dpstp) PROCEDURE (stid);                                           6D
   %called by text editing control routines to accept the stid of a
   bugged statement as inpt and return an appropriate stid for the
   display parameter CDSTOP (an stid on the same level or higher
   which appears after "stid" in the file).%
   UNTIL NOT <FILMNP, getftl> (stid)
      DO stid _ <FILMNP, getsuc> (stid);
   RETURN (stid _ getsuc (stid));
   END.
```

FINISH

```
(strbug) PROCEDURE(ptr);                                                9A
   %this routine will store the current bug mark in the pointer
   passed it.%
   %-------------%
   REF ptr;
   IF bugreg = endfil THEN
      ptr _ pbug(lccords() : ptr[1])
   ELSE
      BEGIN
      ptr _ bugreg;
      ptr[1] _ bugreg[1];
      END;
   RETURN;
   END.

%Input support%
   (inname) PROCEDURE(astrng);                                          9B1
      CASE innmwd(astrng, FALSE, $nmdr) OF
         =0:
            RETURN;
         =1:
            BEGIN
            BUMP [m];
            RETURN;
            END;
         ENDCASE
            BEGIN
            BUMP [m],[m];
            RETURN;
            END;
         END.

   (innmwd)                                                             9B2
      %This routine reads characters from the work station, appends
      them to a register, and displays them in the nameregister.
      Alphabetic characters are forced to upper case before
      insertion into the A-string if the wordflag is false.  The
      argument should be the address of the register into which the
      A-string is to be put. (Note that this routine does not clear
      the A-string before appending characters to it.).%
      %---------------%
      PROCEDURE(astrng, wordflag, delimproc);
      LOCAL char;
      REF astrng, delimproc;
      af();
      disarm();
      LOOP
         BEGIN
         dn(&astrng);
         CASE char _ lookc() OF
            =CD:
               BEGIN
               input();
               GOTO STATE;
               END;
            =CA, =C.:
```

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

%  Replaced by builtin in CLI
(oldanswer) %%this procedure accepts a yes or no answer from the
keyboard, and returns with a 0 if the answer was negative, and a
1 if it was positive%%                                          9B4
   %%------------------%%
   PROCEDURE;
   CASE nlmode OF
      = fulldisplay:
         CASE inpcuc() OF
            = 'Y, = CA, = SP:
               BEGIN
               DSP(?OFF Yes);
               RETURN(TRUE);
               END;
            = 'N:
               BEGIN
```

```
                DSP(?OFF No);
                RETURN(FALSE)
                END;
            = CD: GOTO STATE;
            ENDCASE
                BEGIN
                qm();
                REPEAT;
                END;
        = typewriter:
            BEGIN
            echoOff();
            CASE inpcuc() OF
                = 'Y, =SP, = CA:
                    BEGIN
                    echo($"Yes ");
                    curchr _ 'Y;
                    RETURN(TRUE);
                    END;
                = 'N:
                    BEGIN
                    echo($"No ");
                    curchr _ 'N;
                    RETURN(FALSE)
                    END;
                = CD: GOTO STATE;
                = '?:
                    BEGIN
                    typeas($"
                    Type CD to abort; 'y, SP, or CA for YES; 'n for
                    NO: ");
                    REPEAT;
                    END;
                ENDCASE
                    BEGIN
                    typeas($" ?? ");
                    REPEAT;
                    END;
            END;
        ENDCASE err($"Illegal parsing mode detected in ANSWER");
    END.

%
```

```
< NINE, BCONST.NLS;38, >, 15-Aug-78 10:50 HGL ;;;;

FILE bconst % (arcsubsys,xL10,) (arcsubsys,1109,) (RELNINE,bconst.rel,)
%

% MAP OF NLS SYSTEM
   Pages   0 -- 377
      Low segment of code is loaded up from page 0; the symbol table is
      loaded down from page 400.  If they overlap, we are in trouble.
      (Also, the loader someties blows up in mysterious ways in this
      case!)  When a user program is loaded, a mark is placed at the
      end of the symbol table and the user program symbol's are placed
      down from that.  Thus it pays to have some room there.  The code
      in this segment is generally core routines.
   Pages 400 -- 551
      High segment is used for the CML parser and interpreter code as
      well as parse support code (X- routines, etc.)  When processors
      (compilers, output processors) are run, this code is swapped out
      and the processors mapped into this area.  Upon completion, this
      code is brought back.  See (nls, seqfil, processor).
   Page  552
      Used for the user-option system.  Definition of this page is in
      (nls,updata,).
   Page  553
      Used for mapping pages out in Output File.  This and 552 must not
      be overwritten by code!
   Pages 554 -- 653
      Used for the User Program Buffer area (lower part) and for the
      File buffer area (upper part).  The boundary may be moved by the
      user using the Set Buffer command in the Program subsystem.  The
      default size of the Program Buffer is 4 pages.
   Pages 654 -- 657
      Currently unused.  May be used to extend the size of the loader
      hash table area.
   Pages 660 -- 677
      Loader hash table area.  Filled from 677777B down.
   Page  700
      Used by the loader.
   Pages 701 -- 704                                                    2I
      Sequence stack area. Two stacks of 2 pages each                  2I1
   Pages 705 -- 757                                                    2J
      Storage allocator space. 53 Octal (43 decimal) pages used for
      isrt tables, string storage, sequence generator stacks, and
      sequence generator statement vector work areas.                  2J1
   Pages 760 -- 764
      Loader.
   Pages 765 -- 767
      Currently unused.
   Pages 770 -- 777
      DDT.
   %

% Locations in high core %
   (sqstks)     EXTERNAL ADDRESS = 701000B;                            3A
      % 2 Sequence stacks of 2000B each to 704777B %
   (dspblk)     EXTERNAL ADDRESS = 705000B;                            3B
```

```
    % beginning of storage allocator block; used to store lsrt's,
    strings from getstring, etc. %
  (dspbke)      EXTERNAL ADDRESS = 757777B;                          3C
    % end of storage allocator block %
  (bfree)       EXTERNAL ADDRESS = 400000B;                          3D
  (efree)       EXTERNAL ADDRESS = 554000B;                          3E
    % these pages are currently used for:
       compilers
       Output Processor
       Update Checkpoint, etc.
       load 940 files
       Journal Delivery
       Sort/Merge %
  (ckpag)       EXTERNAL ADDRESS = 553000B;                          3F
    % page used in updtfl as work area %
% Size of storage allocation block: 57 octal (47 decimal) pages %
  (dpybks)      EXTERNAL CONSTANT = 53000B;                          4A
% Character codes for special characters %
  (ascbst)      EXTERNAL ADDRESS = 21B;  % backspace stmt %          5A
  (asctsw)      EXTERNAL ADDRESS = 36B;  % text switch %             5B
  (ctla)        EXTERNAL ADDRESS = 1;    % control a %               5C
  (ctld)        EXTERNAL ADDRESS = 4;    % control d %               5D
  (ctle)        EXTERNAL ADDRESS = 5;    % control e %               5E
  (ctlf)        EXTERNAL ADDRESS = 6;    % control f %               5F
  (ctlg)        EXTERNAL ADDRESS = 7;    % control g %               5G
  (ctln)        EXTERNAL ADDRESS = 16B;  % control n %               5H
  (ctlo)        EXTERNAL ADDRESS = 17B;  % control o %               5I
  (ctlp)        EXTERNAL ADDRESS = 20B;  % control p %               5J
  (ctlr)        EXTERNAL ADDRESS = 22B;  % control r %               5K
  (ctls)        EXTERNAL ADDRESS = 23B;  % control s %               5L
  (ctlt)        EXTERNAL ADDRESS = 24B;  % control t %               5M
  (ctlu)        EXTERNAL ADDRESS = 25B;  % control u %               5N
  (ctlv)        EXTERNAL ADDRESS = 26B;  % control v %               5O
  (ctlw)        EXTERNAL ADDRESS = 27B;  % control w %               5P
  (ctlx)        EXTERNAL ADDRESS = 30B;  % control x %               5Q
  (ctly)        EXTERNAL ADDRESS = 31B;  % control y %               5R
  (ctlz)        EXTERNAL ADDRESS = 32B;  % control z %               5S
  (ascalt)      EXTERNAL ADDRESS = 33B;  % altmode %                 5T
% Group names, for use with define state pop (ds) %
  (spec)        EXTERNAL ADDRESS = 1;    % special group %           6A
  (edit)        EXTERNAL ADDRESS = 2;    % edit group %              6B
  (jump)        EXTERNAL ADDRESS = 3;    % jump group %              6C
  (vect)        EXTERNAL ADDRESS = 4;    % vector package group %    6D
% File things %
   % For global symbolic constants %
     (rfpmax)   EXTERNAL CONSTANT = 100B;% number core pages for file
     blocks %                                                        7A1
     (rngbas)   EXTERNAL CONSTANT = 6;  % first file page for ring
     blocks %                                                        7A2
     (rngm)     EXTERNAL CONSTANT = 95; % max number of ring blocks %
                                                                     7A3
     (dtbbas)   EXTERNAL CONSTANT = 101;        % first file page for
     data blocks %                                                   7A4
     (dtbm)     EXTERNAL CONSTANT = 370;        % max number of data
     blocks %                                                        7A5
     (ringl)    EXTERNAL CONSTANT = 5;  % length of ring element %   7A6
```

```
    (fbhdl)   EXTERNAL CONSTANT = 2;  % length of file block header %
                                                                    7A7
    (sworkl)  EXTERNAL CONSTANT = 5;  % length of work area for
    fechcl %                                                        7A8
    (hdtyp)   EXTERNAL CONSTANT = 0;  % block type for header blocks
    %                                                               7A9
    (sdbtyp)  EXTERNAL CONSTANT = 1;  % block type for sdb blocks %
                                                                    7A10
    (rngtyp)  EXTERNAL CONSTANT = 2;  % block type for ring blocks %
                                                                    7A11
    (jnktyp)  EXTERNAL CONSTANT = 3;  % block type for misc blocks %
                                                                    7A12
    (niltyp)  EXTERNAL CONSTANT = 4;  % block type for blocks to be
    init'ed %                                                       7A13
    (blksiz)  EXTERNAL CONSTANT = 512;        % random file block
    size %                                                          7A14
% Data block property types %
    % Users wishing to have special properties in their programs need
    not declare the properties here unless they are to be
    incorporated into the NLS system or unless they wish to avoid
    conflict with other special applications.  The properties
    reserved for such experimental user applications are
    40000B-77777B.  Declared properties may be used for other
    purposes only if the programmer is sure there will be no conflict
    (such is currently the case with QUERY3 making use of some of the
    graphics types.)  This practice is NOT recommended!  If a
    property does not fall in the USER range, it must be assigned an
    order through inclusion in the table PROPTAB.  See (nine, filmnp,
    locprop) and (nine, filmnp, getptab). %
        (txttyp)        EXTERNAL CONSTANT = 0;  % text data block % 7B1A
        (gtftyp)        EXTERNAL CONSTANT = 1;  % graphics text format
        data block %                                                7B1B
        (lwtyp)         EXTERNAL CONSTANT = 2;  % line work data block
        %                                                           7B1C
        (chtyp)         EXTERNAL CONSTANT = 3;  % cell header data
        block %                                                     7B1D
        (dhtyp)         EXTERNAL CONSTANT = 4;  % diagram header data
        block %                                                     7B1E
% Data block property location table %
    (proptab) EXTERNAL _ (                                          7C1
        5,  % length of property table -- number of properties %
        1,  % gtftyp %
        0,  % txttyp %
        2,  % lwtyp %
        3,  % chtyp %
        4); % dhtyp %
    % defines the order in which system properties will occur in the
    statement.  Used to prevent the necessity of searches of all
    properties.  Need not be used for special user properties.  See
    comment above with the definition of the property types. %
% STID of origin with stfile = 0 %
    (orgstid) EXTERNAL CONSTANT = 2;                                7D1
% Core locations for file block pages %
    (crpgad)  EXTERNAL _ (   % indexed by numbers in [rfpmin,rfpmax] 
    %                                                               7E1
        % Filler %
```

```
        0,
      % 100B pages in high core %
        554000B, 555000B, 556000B, 557000B,
        560000B, 561000B, 562000B, 563000B,
        564000B, 565000B, 566000B, 567000B,
        570000B, 571000B, 572000B, 573000B,
        574000B, 575000B, 576000B, 577000B,
        600000B, 601000B, 602000B, 603000B,
        604000B, 605000B, 606000B, 607000B,
        610000B, 611000B, 612000B, 613000B,
        614000B, 615000B, 616000B, 617000B,
        620000B, 621000B, 622000B, 623000B,
        624000B, 625000B, 626000B, 627000B,
        630000B, 631000B, 632000B, 633000B,
        634000B, 635000B, 636000B, 637000B,
        640000B, 641000B, 642000B, 643000B,
        644000B, 645000B, 646000B, 647000B,
        650000B, 651000B, 652000B, 653000B);
% Directories %
   % (prtdir)        EXTERNAL STRING _ "ARCPRINTER";  moved to
   uodata %
   (comdir)  EXTERNAL STRING _ "COM";                                  7F2
% Miscellaneous %
   (nvtk)    EXTERNAL CONSTANT = 2;                                    7G1
   (n40fil)  EXTERNAL CONSTANT = 0;                                    7G2
   (tenfil)  EXTERNAL CONSTANT = 1;                                    7G3
   (macfil)  EXTERNAL CONSTANT = 2;                                    7G4
   (assfil)  EXTERNAL CONSTANT = 3;                                    7G5
   (heurfil) EXTERNAL CONSTANT = 88;                                   7G6
   (justfil) EXTERNAL CONSTANT = 89;                                   7G7
   (opsqfl)  EXTERNAL CONSTANT = 1;                                    7G8
   (opqpfl)  EXTERNAL CONSTANT = 2;                                    7G9
   (opmcfl)  EXTERNAL CONSTANT = 3;                                   7G10
   (opmlfl)  EXTERNAL CONSTANT = 4;                                   7G11
   (bintyp)  EXTERNAL CONSTANT = 2;                                   7G12
   (chrtyp)  EXTERNAL CONSTANT = 3;                                   7G13
   (comtyp)  EXTERNAL CONSTANT = 5;                                   7G14
   (lpttype) EXTERNAL CONSTANT = 6;                                   7G15
   (netype)  EXTERNAL CONSTANT = 7;                                   7G16
   (chkpcf)  EXTERNAL CONSTANT = TRUE;                                7G17
   (origff)  EXTERNAL CONSTANT = FALSE;                               7G18
   (oldvrsn) EXTERNAL CONSTANT = -4;                                  7G19
   (dfltvrs) EXTERNAL CONSTANT = FALSE;                               7G20
% File types %
   (pspublic)        EXTERNAL ADDRESS = 1;                             7H1
   (psprivate)       EXTERNAL ADDRESS = 2;                             7H2
% Miscellaneous %
   (acchecked)       EXTERNAL;                                         7I1
   (skpachk)         EXTERNAL CONSTANT = FALSE;                        7I2
   (lgngrps)         EXTERNAL CONSTANT = 0;                            7I3
% extensions %
   (nlsext)          EXTERNAL STRING _ "NLS";                          7J1
   (savext)          EXTERNAL STRING _ "SAV";                          7J2
   (exeext)          EXTERNAL STRING _ "EXE";                          7J3
   (ctlext)          EXTERNAL STRING _ "CTL";                          7J4
   (relext)          EXTERNAL STRING _ "REL";                          7J5
```

```
   (txtext)          EXTERNAL STRING _ "TXT";                          7J6
   (cgrext)          EXTERNAL STRING _ "CGR";                          7J7
   (subext) EXTERNAL STRING _ "SUBSYS";                                7J8
   (caext)           EXTERNAL STRING _ "CA";                           7J9
   (skext)  EXTERNAL STRING _ "SK";                                   7J10
   (sgext)  EXTERNAL STRING _ "SG";                                   7J11
   (procext)EXTERNAL STRING _ "PROC-REP";                             7J12
% flag values for GTJFN JSYS  %
   (gtjoof)          EXTERNAL CONSTANT = 4B11; % output file
   (default version next higher) %                                    7K1
   (gtjoif)          EXTERNAL CONSTANT = 1B11; % old file only %      7K2
   (gtjprf)          EXTERNAL CONSTANT = 1B11; % old file only %      7K3
   (gtjoosf)         EXTERNAL CONSTANT = 4B11; % output file
   (default version next higher) %                                    7K4
   (gtjoisf)         EXTERNAL CONSTANT = 1B11; % old file only %      7K5
   (gtjstr)          EXTERNAL CONSTANT = 1B8; % accept file group
   descriptor, determining name according to old/new file bits %      7K6
   (gtjprv)          EXTERNAL CONSTANT = 2B9; % no access by other
   forks %                                                            7K7
   (gtjidl)          EXTERNAL CONSTANT = 1B9; % ignore deleted bit %
                                                                      7K8
   (gtjnfo)          EXTERNAL CONSTANT = 2B11; % new file only %      7K9
   % applied to LH of flag word by getgtjflg procedure %
      (gtjwrt)             EXTERNAL CONSTANT = 4B5; % output file
      (default next version higher) like 4B11 %                       7K10A
      (gtjred)             EXTERNAL CONSTANT = 1B5; % old file
      only (like 1B11) %                                              7K10B
      (gtjigdel)     EXTERNAL CONSTANT = 1B3; % ignore deleted bit
      (1B9) %                                                         7K10C
      (gtjcpc)             EXTERNAL CONSTANT = 2B3; % no access by
      othe forks (2B9) %                                              7K10D
   (sgtjtf)          EXTERNAL CONSTANT = 1B6; % use short form of
   GTJFN %                                                            7K11
   (jfnstf)          EXTERNAL CONSTANT = 111100400001B; % for JFNS
   (JFN to string) JSYS %                                             7K12
   (jfnnamonly)      EXTERNAL CONSTANT = 1B9;                         7K13
% file access types %
   (write)           EXTERNAL CONSTANT = 0;                           7L1
   (read)            EXTERNAL CONSTANT = 1;                           7L2
   (readwrite)       EXTERNAL CONSTANT = 2;                           7L3
   (append)          EXTERNAL CONSTANT = 3;                           7L4
   (rthawed)         EXTERNAL CONSTANT = 5;                           7L5
   (rwthawed)        EXTERNAL CONSTANT = 4;                           7L6
   (opnmod)  EXTERNAL CONSTANT = 1;                                   7L7
      %access mode for opening original file in 'coropnfil'
      procedure.  'read' for running outside of NSW.  Changed in
      'presave' to 'readwrite' for NSW.%
% Constants from IOEXEC (file interface) %
   (bfile)           EXTERNAL CONSTANT = 4;  % bad file error no %
                                                                      7M1
% Processors %
 % Types of output processors %
   (upgtyp)  EXTERNAL CONSTANT = 1;  % compile a user program to the
   buffer %                                                           8A1
   (cmptyp)  EXTERNAL CONSTANT = 2;  % compile a program to a REL
   file %                                                             8A2
```

```
    (odtyp)   EXTERNAL CONSTANT = 3;  % output device something %  8A3
  % A call [[prcadr]] gets to processor %
    (prcadr)  EXTERNAL CONSTANT = 400010B;                        8B1
% Type codes %
  % Update %
    (newversion)      EXTERNAL CONSTANT = 1;                      9A1
    (oldversion)      EXTERNAL CONSTANT = 2;                      9A2
    (upcompact)       EXTERNAL CONSTANT = 3;                      9A3
    (newname)         EXTERNAL CONSTANT = 4;                      9A4
  % Structure manipulation %
    (copyflag)        EXTERNAL CONSTANT = 1;                      9B1
    (moveflag)        EXTERNAL CONSTANT = 2;                      9B2
    (trnsflag)        EXTERNAL CONSTANT = 3;                      9B3
    (deltflag)        EXTERNAL CONSTANT = 4;                      9B4
% Terminal stuff %
  % System type %
    (dntyp)           EXTERNAL CONSTANT = 0;  % user running display
    NLS %                                                         10A1
    (tntyp)           EXTERNAL CONSTANT = 1;  % user running
    typewriter NLS %                                              10A2
    (nttyp)           EXTERNAL CONSTANT = 2;                      10A3
  % Device types %
    (arcoprtty)       EXTERNAL CONSTANT = 27B;        % ARC
    operator's tty number %                                       10B1
    (utiloprtty)      EXTERNAL CONSTANT = 1B; % utility operator's
    tty number%                                                   10B2
    (nsaoprtty)       EXTERNAL CONSTANT = 0B; % NSA operator's tty
    number%                                                       10B3
  % Systems cinits %
    (nwheecinits)     EXTERNAL _ (                                10C1
       1210400B, 320440B,  3061400B, 1016600B,
       1313300B, 526640B,  1324640B, 1213340B,
       1205000B, 1207000B, 1323113B,  446640B,
       2230133B, 0,0,0,0,0,0,0,0,0);
  % File access bit definitions %
    (racc)    EXTERNAL CONSTANT = 100000B6;   % read access (bit 2) %
                                                                  10D1
    (wacc)    EXTERNAL CONSTANT =  40000B6;   % write access (bit 3)
    %                                                             10D2
    (eacc)    EXTERNAL CONSTANT =  20000B6;   % execute access (bit
    4) %                                                          10D3
    (tuacc)   EXTERNAL CONSTANT =   1000B6;   % trap to user on
    access (b 8) %                                                10D4
    (cwacc)   EXTERNAL CONSTANT =    400B6;   % copy on write access
    (bit 9) %                                                     10D5
% Sequence Generator %
  % Tells user sequence generators what they are being called as %
    (sqopn)   EXTERNAL CONSTANT = 1;                              11A1
    (sqcls)   EXTERNAL CONSTANT = 2;                              11A2
    (sqgnxt)  EXTERNAL CONSTANT = 3;                              11A3
    (sqstkn) EXTERNAL CONSTANT = 2; % number of sequence work areas
    allocated at initialization %                                 11A4
% Values for modes %
  % All multiply used values should be defined here %
    (vexpert)         EXTERNAL ADDRESS = 1;   % expert %          12A1
    (vfullprompts)    EXTERNAL ADDRESS = 3;   % full prompting %  12A2
```

```
    (vtersemode)      EXTERNAL ADDRESS = 1;   % terse mode %       12A3
    (vmultchar)       EXTERNAL ADDRESS = 2;   % mult char hearald %
                                                                   12A4
    (vverbsmode)      EXTERNAL ADDRESS = 2;   % verbose feedback %
                                                                   12A5
% Record size declarations %
  % Length of pvsrecord (offset to viewspec collection astring) %
    (pvslen)  EXTERNAL CONSTANT = 3;                               13A1
  % Length of plvrecord (offset to leveladj collection astring) %
    (plvlen)  EXTERNAL CONSTANT = 1;                               13B1
  % Length of pconrecord %
    (pconlen) EXTERNAL CONSTANT = 1;                               13C1
  % Length of pkeyrecord %
    (pkeylen) EXTERNAL CONSTANT = 1;                               13D1
% Level adjust value codes %
  (levsuc)     EXTERNAL CONSTANT = 0;  % value for successor %      14A
  (levup)      EXTERNAL CONSTANT = 1;  % value for up one level %   14B
  (levdown)    EXTERNAL CONSTANT = -1; % value for down one level % 14C
% Description of the data structure for parsed links %
  (lfn)        EXTERNAL CONSTANT = 0;  % file number for default
  directory %                                                      15A
  (ls)         EXTERNAL CONSTANT = 1;  % text pointer to get start of
  link %                                                           15B
  (le)         EXTERNAL CONSTANT = 3;  % text pointer to get end of
  link %                                                           15C
  (cs)         EXTERNAL CONSTANT = 5;  % text pointer to get start of
  comment %                                                        15D
  (ce)         EXTERNAL CONSTANT = 7;  % text pointer to get end of
  comment %                                                        15E
  (hs)         EXTERNAL CONSTANT = 9;  % text pointer to get start of
  hostname%                                                        15F
  (he)         EXTERNAL CONSTANT = 11; % text pointer to get end of
  hostname %                                                       15G
  (us)         EXTERNAL CONSTANT = 13; % text pointer to get start of
  username%                                                        15H
  (ue)         EXTERNAL CONSTANT = 15; % text pointer to get end of
  username %                                                       15I
  (fs)         EXTERNAL CONSTANT = 17; % text pointer to get start of
  filename%                                                        15J
  (fe)         EXTERNAL CONSTANT = 19; % text pointer to get end of
  filename %                                                       15K
  (ds)         EXTERNAL CONSTANT = 21; % text pointer to get start of
  dae %                                                            15L
  (de)         EXTERNAL CONSTANT = 23; % text pointer to get end of
  dae %                                                            15M
  (vb)         EXTERNAL CONSTANT = 25; % text pointer to get start of
  viewspc %                                                        15N
  (ve)         EXTERNAL CONSTANT = 27; % text pointer to get end of
  viewspecs %                                                      15O
  (lnkdsl)     EXTERNAL CONSTANT = 29; % length of this data structure
  %                                                                15P
% User-options data page layout definitions %
  % Declarations for the user-profile page are in file UPDATA %
% Default and maximum size for user programs buffer%
  (upgbdf)     EXTERNAL ADDRESS = 6;                               17A
  (maxbsize)   EXTERNAL CONSTANT = 44;                             17B
```

```
   (minfpages)  EXTERNAL CONSTANT = 20;                                  17C
% PROGRAMS subsys constants %
   % globals strings %
      (cprcdname) EXTERNAL STRING = "$COMPILES"; %sequential recording
      filename%                                                          18A1
      (l10link) EXTERNAL STRING = "<ARCSUBSYS,L109,>";                   18A2
      (cmllink) EXTERNAL STRING = "<ARCSUBSYS,CML10,>";                  18A3
% Insert Sendmail form elements %                                        19
   (sjaccess) EXTERNAL STRING _ "ACCESS STATUS:";                        19A
   (sjaction) EXTERNAL STRING _ "DISTRIBUTE FOR ACTION TO:";             19B
   (sjauthor) EXTERNAL STRING _ "AUTHOR(S):";                            19C
   (sjbranch) EXTERNAL STRING _ "BRANCH AT:";                            19D
   (sjcomments) EXTERNAL STRING _ "COMMENT:";                            19E
   (sjfile) EXTERNAL STRING _ "FILE:";                                   19F
   (sjforward) EXTERNAL STRING _ "FORWARD ITEM NUMBER:";                 19G
   (sjgroup) EXTERNAL STRING _ "GROUP AT:";                              19H
   (sjhandle) EXTERNAL STRING _ "HANDLING INSTRUCTION:";                 19I
   (sjhardcopy) EXTERNAL STRING _ "OFFLINE ITEM -- LOCATED AT:";         19J
   (sjinfo) EXTERNAL STRING _ "DISTRIBUTE FOR INFO-ONLY TO:";            19K
   (sjkeyword) EXTERNAL STRING _ "KEYWORD(S):";                          19L
   (sjlink) EXTERNAL STRING _ "INSERT LINK TO FOLLOW:";                  19M
   (sjmessage) EXTERNAL STRING _ "MESSAGE:";                             19N
   (sjnumber) EXTERNAL STRING _ "NUMBER:";                               19O
   (sjobsolete) EXTERNAL STRING _ "OBSOLETES ITEM NUMBER(S):";           19P
   (sjplex) EXTERNAL STRING _ "PLEX AT:";                                19Q
   (sjrecord) EXTERNAL STRING _ "RECORDING INSTRUCTION:";                19R
   (sjrfc) EXTERNAL STRING _ "RFC NUMBER:";                              19S
   (sjsend) EXTERNAL STRING _ "SEND THE MAIL.";                          19T
   (sjsubcol) EXTERNAL STRING _ "SUBCOLLECTION(S):";                     19U
   (sjtitle) EXTERNAL STRING _ "TITLE:";                                 19V
   (sjupdate) EXTERNAL STRING _ "UPDATE TO ITEM NUMBER(S):";             19W
% Collecter sorter %
   (v)          EXTERNAL ADDRESS = 400000B;  % address of sort vector %
                                                                         20A
   (vcvu)       EXTERNAL ADDRESS = 400100B;                              20B
   (vcvul)      EXTERNAL ADDRESS = 400070B;                              20C
   (vcv)        EXTERNAL ADDRESS = 420000B;  % primary sort key value %
                                                                         20D
   (vmax)       EXTERNAL CONSTANT = 20000B;     % max length of sort
   vector %                                                              20E
   (vcvend)     EXTERNAL CONSTANT = 477750B;    % end of sort key value
   area %                                                                20F
% Level adjust parameters %
   (downstr)    EXTERNAL STRING _ "d";                                   21A
   (sucstr)     EXTERNAL STRING [0];                                     21B
   (sucdir)     EXTERNAL CONSTANT = 0;                                   21C
   (down)       EXTERNAL CONSTANT = -1;                                  21D
% Structure type definitions %
   (stmtv)      EXTERNAL CONSTANT = 1;                                   22A
   (brnchv)     EXTERNAL CONSTANT = 2;                                   22B
   (groupv)     EXTERNAL CONSTANT = 3;                                   22C
   (plexv)      EXTERNAL CONSTANT = 4;                                   22D
   (litv)       EXTERNAL CONSTANT = 5;                                   22E
   (filev)      EXTERNAL CONSTANT = 6;                                   22F
   (hcopyv)     EXTERNAL CONSTANT = 7;                                   22G
   (charv)      EXTERNAL CONSTANT = 8;                                   22H
```

```
   (wordv)      EXTERNAL CONSTANT = 9;                                  22I
   (numbrv)     EXTERNAL CONSTANT = 10;                                 22J
   (visv)       EXTERNAL CONSTANT = 11;                                 22K
   (invisv)     EXTERNAL CONSTANT = 12;                                 22L
   (linkv)      EXTERNAL CONSTANT = 13;                                 22M
   (textv)      EXTERNAL CONSTANT = 14;                                 22N
   (tdfltv)     EXTERNAL CONSTANT = 15; % for substitute %              22O
   (forwdv)     EXTERNAL CONSTANT = 16; % for journal forwarding of
   mail %                                                               22P
% selection types %                                                     23
   (lseltyp) EXTERNAL CONSTANT = 1;                                     23A
   (sseltyp) EXTERNAL CONSTANT = 0;                                     23B
   (cwstring) EXTERNAL CONSTANT = 100;                                  23C
   (cwstructure) EXTERNAL CONSTANT = 101;                               23D
   (cweditent) EXTERNAL CONSTANT = 0;                                   23E
% File type definitions %
   (random)     EXTERNAL CONSTANT = 4;                                  24A
% Lookup types %
   (nametyp)    EXTERNAL CONSTANT = 1;  % fast name lookup from start
   of file %                                                            25A
   (wordtyp)    EXTERNAL CONSTANT = 2;  % word search %                 25B
   (contnt)     EXTERNAL CONSTANT = 3;  % content search %              25C
   (sid)        EXTERNAL CONSTANT = 4;  % sid search %                  25D
   (contls)     EXTERNAL CONSTANT = 6;  % single statement content
   search %                                                             25E
   (nxtname)    EXTERNAL CONSTANT = 7;  % fast name lookup from
   specified STID %                                                     25F
   (segname)    EXTERNAL CONSTANT = 8;  % sequential name lookup %     25G
   (braname)    EXTERNAL CONSTANT = 9;  % name lookup within specified
   branch %                                                             25H
   (extname)    EXTERNAL CONSTANT = 10; % external name - take link in
   sysgd %                                                              25I
   (wordls)     EXTERNAL CONSTANT = 11; % single statement word lookup
   %                                                                    25J
% Control Character echoing control %
   (ttycoc)     EXTERNAL CONSTANT = 10737B3;                            26A
   (dpycoc)     EXTERNAL CONSTANT = 104012537B3;                        26B
% Space for tab values.  See printoptions for tabstops %
   (defspftab)  EXTERNAL CONSTANT =  0;
   (defrtjtab)  EXTERNAL CONSTANT =  0;
   (defrjtchr)  EXTERNAL CONSTANT = 11B;
% Subsystem names %
   (opname)     EXTERNAL STRING _ "<NETSYS>OUTPRC.SAV";                 28A
   (xopname)    EXTERNAL STRING _ "<NINEPORGEN>XOUTPRC.SAV";            28B
   %(idnfname) EXTERNAL STRING _ "<IDENTS>IDENTS.NLS";% %changed by
   (bintnls,saveit) for some hosts%  %moved to BDATA%
   (jnlnam)     EXTERNAL STRING _ "JOURNL";                             28D
   % (trenam)   EXTERNAL STRING _ "TREEMETA"; %
   (subdir)     EXTERNAL STRING _ "NETSYS";                             28F
   (subdr2)    EXTERNAL STRING _ "SUBSYS";
   %(subdfdir)  EXTERNAL STRING [40];%  % default directory for
   programs (subsystems too) %  %moved to BDATA%
% Identification System %
   (indtyp)     EXTERNAL CONSTANT = 1;                                  29A
   (grptyp)     EXTERNAL CONSTANT = 2;                                  29B
   (orgtyp)     EXTERNAL CONSTANT = 3;                                  29C
```

```
   (afgtyp)     EXTERNAL CONSTANT = 4;                                  29D
% Device things %
   % Devices for Output Processor %
      (optydv)  EXTERNAL CONSTANT = 2;  % (local) terminal %          30A1
      (opcmdv)  EXTERNAL CONSTANT = 3;  % COM %                      30A2
      (opprdv)  EXTERNAL CONSTANT = 4;  % Printer %                  30A3
      (opxpdv)  EXTERNAL CONSTANT = 6;  % COM test %                 30A4
      (opxtdv)  EXTERNAL CONSTANT = 7;  %  (local) terminal  COM test %
                                                                      30A5
      (oprmdv)  EXTERNAL CONSTANT = 10; % remote Printer/Terminal % 30A6
      (optyfl)  EXTERNAL CONSTANT = 11; % terminal file %           30A7
% Imlac protocol %
   % Control codes for remote displays (e.g. imlacs) %
      (remada)  EXTERNAL CONSTANT = 1;  % alocate display area control
      char %                                                          31A1
      (remdda)  EXTERNAL CONSTANT = 2;  % dealocate display area
      control char %                                                  31A2
      (remstr)  EXTERNAL CONSTANT = 4;  % string manipulation control
      char %                                                          31A3
      (remscsr) EXTERNAL CONSTANT = 5;  % set cursor string control
      char %                                                          31A4
      (remssda) EXTERNAL CONSTANT = 10B;        % suppress string in da
      control char %                                                  31A5
      (remscm)  EXTERNAL CONSTANT = 15B;        % switch to short char
      mode %                                                          31A6
      (nwada)   EXTERNAL CONSTANT = 16B;        % new ada %         31A7
      (apsda)   EXTERNAL CONSTANT = 17B;        % append string to seq
      display area %                                                  31A8
      (nwstrda) EXTERNAL CONSTANT = 20B;        % new strda %       31A9
      (ccnda)   EXTERNAL CONSTANT = 21B;        % start contin trans of
      cursor coords %                                                 31A10
      (ccfda)   EXTERNAL CONSTANT = 22B;        % stop contin trans of
      cursor coords %                                                 31A11
      (reminit) EXTERNAL CONSTANT = 33B;        % initialize all da
      control char %                                                  31A12
   (remfudge)   EXTERNAL CONSTANT = 40B;                               31B
   (begmsg)     EXTERNAL CONSTANT = 33B;        % begin message char %
                                                                      31C
   (remsdda)    EXTERNAL CONSTANT = 6;  % suppress display area control
   char %                                                             31D
   (remrdda)    EXTERNAL CONSTANT = 7;  % restore display area control
   char %                                                             31E
   (remtsn)     EXTERNAL CONSTANT = 12B;       % tty simulation on
   control char %                                                     31F
   (remtsf)     EXTERNAL CONSTANT = 13B;       % tty simulation off
   control char %                                                     31G
% PCP-related constants %
   % Indices into selection lists from the frontend%
      (cwtype)  EXTERNAL CONSTANT = 1;                                32A1
      (tppair)  EXTERNAL CONSTANT = 2;                                32A2
      (xcoord)  EXTERNAL CONSTANT = 1;                                32A3
      (ycoord)  EXTERNAL CONSTANT = 2;                                32A4
      (wndw)    EXTERNAL CONSTANT = 3;                                32A5
   % Flags for cprint to do co-routine call to return result or a
   regular return %
      (cortn)   EXTERNAL CONSTANT = 1;                                32B1
```

```
    (regrtn)  EXTERNAL CONSTANT = 0;                                    32B2
    % Length of maximum charstr passed as dps argument %
    (maxdpschar)      EXTERNAL CONSTANT = 1750;                         32C1
    % Flags for PCP call to Frontend SHOW %
    (nocawait)        EXTERNAL CONSTANT = 0;                            32D1
    (cawait)          EXTERNAL CONSTANT = 1;                            32D2
    (cawtnomes) EXTERNAL CONSTANT = 2;                                  32D3
    % PCP836 type codes %
    % Device constants %
    (typewriter)      EXTERNAL CONSTANT = 0;                            32F1
    (fulldisplay)     EXTERNAL CONSTANT = 1;                            32F2
% Display stuff %
    (fontmax)    EXTERNAL CONSTANT = 512;                               33A
    (fnormal)    EXTERNAL CONSTANT = 0;                                 33B
    (rtmax)      EXTERNAL CONSTANT = 60;                                33C
    (dspallf)    EXTERNAL CONSTANT = 6;                                 33D
    % Split screen constants %
    (horizontal)      EXTERNAL CONSTANT = 0;                            33E1
    (vertical)        EXTERNAL CONSTANT = 1;                            33E2
    (minww)           EXTERNAL CONSTANT = 20; % minimum window width
    %                                                                   33E3
    (minwh)           EXTERNAL CONSTANT = 2;  % minimum window heigth
    %                                                                   33E4
    % Protocol codes and other declarations %
      % Line processor protocol codes %
        % Input semantic codes in big characters %
          (fcor2)     EXTERNAL CONSTANT = 42B;      % 2 coords not
          with a char %                                          33F1A1
          (fcor4)     EXTERNAL CONSTANT = 44B;      % 4 coords not
          with a char %                                          33F1A2
          (cnch2)     EXTERNAL CONSTANT = 41B;      % control char
          %                                                      33F1A3
          (acor2)     EXTERNAL CONSTANT = 43B;      % 2 coords with
          N.B. or C.C. %                                         33F1A4
          (acor4)     EXTERNAL CONSTANT = 45B;      % 4 coords with
          a char %                                               33F1A5
          (irep)      EXTERNAL CONSTANT = 46B;      % interrogate
          response %                                             33F1A6
        (lpesc)       EXTERNAL CONSTANT = 33B;      % escape code
        to imlac/LP %                                            33F1B
        (lpresc)      EXTERNAL CONSTANT = 34B;      % escape code
        from imlac/LP %                                          33F1C
        (lpposition)  EXTERNAL CONSTANT = 40B;      % position code
        %                                                        33F1D
        (lptty)       EXTERNAL CONSTANT = 41B;      % specify tty
        window code %                                            33F1E
        (lptrack)     EXTERNAL CONSTANT = 42B;      % resume
        tracking code %                                          33F1F
        (lpcline)     EXTERNAL CONSTANT = 43B;      % clear line
        code %                                                   33F1G
        (lpdline)     EXTERNAL CONSTANT = 44B;      % delete line
        code %                                                   33F1H
        (lpinline)    EXTERNAL CONSTANT = 45B;      % insert line
        code %                                                   33F1I
        (lpbug)       EXTERNAL CONSTANT = 46B;      % bug selection
        code %                                                   33F1J
```

```
    (lppbug)        EXTERNAL CONSTANT = 47B;        % pop bug
    selection code %                                                33F1K
    (lpcscreen)     EXTERNAL CONSTANT = 50B;        % clear screen
    code %                                                          33F1L
    (lpreset)       EXTERNAL CONSTANT = 51B;        % reset code %
                                                                    33F1M
    (lpintter)      EXTERNAL CONSTANT = 55B;        % interrogate
    code %                                                          33F1N
    (lpstandout)    EXTERNAL CONSTANT = 56B;        % stand out
    code %                                                          33F1O
    (lpendstandout)EXTERNAL CONSTANT = 57B;         % end stand out
    code %                                                          33F1P
    (lpnocoor)      EXTERNAL CONSTANT = 60B;        % exit
    coordinate mode %                                               33F1Q
    (lpcoord)       EXTERNAL CONSTANT = 61B;        % enter
    coordinate mode %                                               33F1R
    (lptptopen)     EXTERNAL CONSTANT = 53B;        % LP terminal
    printer open %                                                  33F1S
    (lptptclose)    EXTERNAL CONSTANT = 54B;        % LP terminal
    printer close %                                                 33F1T
    (lptptsr)       EXTERNAL CONSTANT = 52B;        % LP terminal
    printer string %                                                33F1U
% Jump return ring sizes %
   (frrmax)  EXTERNAL CONSTANT = 25; % max entries in file return
   ring %                                                           33G1
   (srrmax)  EXTERNAL CONSTANT = 25; % max entries in statement
   return ring %                                                    33G2
% Strings used to display non-printing characters - text-pointers in
lsrt point here.  Note that CR and EOL are used only by display
name, and are interpreted properly by literal dlsay and stfrmt. %
   (castr)   EXTERNAL STRING _ "<CA>";                              33H1
   (cdotstr) EXTERNAL STRING _ "<C.>";                              33H2
   (lfdstr)  EXTERNAL STRING _ "<LF>";                              33H3
   (cdstr)   EXTERNAL STRING _ "<CD>";                              33H4
   (bcstr)   EXTERNAL STRING _ "<BC>";                              33H5
   (escstr)  EXTERNAL STRING _ "<ESC>";                             33H6
   (astr)    EXTERNAL STRING _ "<^A>";                              33H7
   (bstr)    EXTERNAL STRING _ "<^B>";                              33H8
   (cstr)    EXTERNAL STRING _ "<^C>";                              33H9
   (estr)    EXTERNAL STRING _ "<^E>";                             33H10
   (fstr)    EXTERNAL STRING _ "<^F>";                             33H11
   (gstr)    EXTERNAL STRING _ "<^G>";                             33H12
   (hstr)    EXTERNAL STRING _ "<^H>";                             33H13
   (kstr)    EXTERNAL STRING _ "<^K>";                             33H14
   (lstr)    EXTERNAL STRING _ "<^L>";                             33H15
   (mstr)    EXTERNAL STRING _ "<CR>";                             33H16
   (nstr)    EXTERNAL STRING _ "<^N>";                             33H17
   (ostr)    EXTERNAL STRING _ "<^O>";                             33H18
   (pstr)    EXTERNAL STRING _ "<^P>";                             33H19
   (qstr)    EXTERNAL STRING _ "<^Q>";                             33H20
   (rstr)    EXTERNAL STRING _ "<^R>";                             33H21
   (sstr)    EXTERNAL STRING _ "<^S>";                             33H22
   (tstr)    EXTERNAL STRING _ "<^T>";                             33H23
   (ustr)    EXTERNAL STRING _ "<^U>";                             33H24
   (vstr)    EXTERNAL STRING _ "<^V>";                             33H25
   (wstr)    EXTERNAL STRING _ "<^W>";                             33H26
```

```
    (xstr)     EXTERNAL STRING _ "<^X>";                                33H27
    (ystr)     EXTERNAL STRING _ "<^Y>";                                33H28
    (zstr)     EXTERNAL STRING _ "<^Z>";                                33H29
    (uslstr)   EXTERNAL STRING _ "<^/>";                                33H30
    (ubrstr)   EXTERNAL STRING _ "<^]>";                                33H31
    (uustr)    EXTERNAL STRING _ "<^^>";                                33H32
    (crstr)    EXTERNAL STRING _ "<CR>";                                33H33
    (enlstr)   EXTERNAL STRING _ "<EOL>";                               33H34
    (tbstr)    EXTERNAL STRING _ "<TAB>";                               33H35
    (nulstr)   EXTERNAL STRING _ "<NUL>";                               33H36
  % Correspondence tables for non-printing character display strings %
  % Codes encountered in user files %
    (npcodes)          EXTERNAL _                                       33J1
       (CA, C., LF, CD, BC, 33B,
       1B, 2B, 3B, 5B, 6B, 7B, 10B, 13B, 14B, 15B, 16B, 17B, 20B,
       21B, 22B, 23B, 24B, 25B, 26B, 27B, 30B, 31B, 32B,
       34B, 35B, 36B, CR, EOL, TAB, 0B);
  % Addresses of strings used to disply those codes %
    (npstrings)        EXTERNAL _                                       33K1
       ($castr, $cdotstr, $lfdstr, $cdstr, $bcstr, $escstr,
       $astr, $bstr, $cstr, $estr, $fstr, $gstr, $hstr, $kstr, $lstr,
       $mstr, $nstr, $ostr, $pstr, $qstr, $rstr, $sstr, $tstr, $ustr,
       $vstr, $wstr, $xstr, $ystr, $zstr,
       $uslstr, $ubrstr, $uustr, $crstr, $enlstr, $tbstr, $nulstr);
  % Size of non-printing char tables %
    (npsize) EXTERNAL CONSTANT = 36;                                    33L1
% Storage management %
  (blksuc)     EXTERNAL CONSTANT = 1;  % addr of successor block in
  freelist %                                                            34A
  (blkprd)     EXTERNAL CONSTANT = 2;  % addr of predecessor block in
  freelist%                                                             34B
  (bhl)        EXTERNAL CONSTANT = 1;  % length of an in-use block
  header %                                                              34C
  (blkoffset)  EXTERNAL CONSTANT = 0;                                   34D
    % add this amount to address returned by getblk to get start of
    block; i.e. past block header and string header %
% Copy and Show Directory storage management %
  (oldpgsz)    EXTERNAL CONSTANT = 0;                                   35A
% Journal System %
  % To be filled in when a system is made %
    (jnlpsw)  EXTERNAL STRING [15];                                     36A1
  % On-line startups %
    (jnltime) EXTERNAL _ (12001705408, 7021334B, 0, 011500B, 012200B,
    012500B, 0, 0, 0, 0, 0, 0, 0, 0);                                   36B1
       %  Format described in <RECFIL>setojtime;
          RECORDS at <UTILTY>ojtime;
          4000010100B, 0, causes no delivery or slinker %
% Network %
  (archost)            EXTERNAL CONSTANT = 150;  %ISIC%                 37A
  (utilhost)           EXTERNAL CONSTANT = 66;   %SRI-KL%               37B
  (nsahost)            EXTERNAL CONSTANT = 65;                          37C
  (rfchost)            EXTERNAL CONSTANT = 66;   %host num having RFC
  system%                                                               37D
  (officel) EXTERNAL CONSTANT = 43;                                     37E
  (isichost)           EXTERNAL CONSTANT = 150;                         37F
  (isidhost)           EXTERNAL CONSTANT = 214;                         37G
```

```
(isiehost)          EXTERNAL CONSTANT = 116;                          37H
(isirlhost)         EXTERNAL CONSTANT = 33;   %NELC%                  37I
(sriai20host) EXTERNAL CONSTANT = 66; %SRI-KL%
(srialkahost) EXTERNAL CONSTANT = 51;
(bbnbhost)          EXTERNAL CONSTANT = 49;                           37L

(arcistr)   EXTERNAL STRING _ "SRI-ARC";                              37N
(utilistr)          EXTERNAL STRING _ "OFFICE-1";                     37O
(utilstr)           EXTERNAL STRING _ "SRI-KL";                       37P
(nsastr)            EXTERNAL STRING _ "NSA";                          37Q

(inremsiteflag)     EXTERNAL CONSTANT = TRUE;                         37S
(outremsiteflag)    EXTERNAL CONSTANT = TRUE;                         37T
% TENEX/TOPS20 dependent %
(tvrdchar) EXTERNAL = '; ;                                            38A
   %changed to '. for tops20 at (nine,bintnls,saveit)%
(cpcldelim) EXTERNAL = '$; %changed to "$" for tops20 (and ISID)%     38B
(cpcrdelim) EXTERNAL = '$;  %changed like cpcldelim%                  38C
(t2fsbegin) EXTERNAL = 3;                                             38D
%file name strings which get changed for tops20 by
(nine,bintnls,saveit)%
   (dirsfname) EXTERNAL STRING = "*.*;*";                             38E1
   (flgfname) EXTERNAL STRING = "<NETSYS>NLSFLAGS.FLAGS;1";           38E2
   (grpfname) EXTERNAL STRING = "<NETSYS>GROUP.INDEX;1";              38E3
(t2fsend) EXTERNAL = 0; %end of filename strings%                     38F
% FTP %
(emrtnfalse) EXTERNAL ADDRESS = 1;                                    39A
(emsignal)   EXTERNAL ADDRESS = 2;                                    39B
(emreset)    EXTERNAL ADDRESS = 3;                                    39C
% Use measurement %
(mstxte)     EXTERNAL CONSTANT = 0;                                   40A
(msstre)     EXTERNAL CONSTANT = 1;                                   40B
(mschrt)     EXTERNAL CONSTANT = 2;                                   40C
(mswsiw)     EXTERNAL CONSTANT = 3;                                   40D
(msstart)    EXTERNAL CONSTANT = 1;                                   40E
(msfinish)   EXTERNAL CONSTANT = 0;                                   40F
(msbkl1)     EXTERNAL CONSTANT = 3;                                   40G
(msbk21)     EXTERNAL CONSTANT = 4;                                   40H
(mshdrl)     EXTERNAL CONSTANT = 2;                                   40I
(msbavp)     EXTERNAL CONSTANT = 30;                                  40J
(msrecl)     EXTERNAL CONSTANT = 35;                                  40K
(msbufl)     EXTERNAL CONSTANT = 512;                                 40L
(msdsys)     EXTERNAL CONSTANT = 0;                                   40M
(mstsys)     EXTERNAL CONSTANT = 1;                                   40N
% Offsets into measurement buffer %
   (msproc)  EXTERNAL ADDRESS = 0;                                    4001
   (msreal)  EXTERNAL ADDRESS = 1;                                    4002
   (mscount) EXTERNAL ADDRESS = 2;                                    4003
   (mspmtot) EXTERNAL ADDRESS = 3;                                    4004
   (mscurt)  EXTERNAL ADDRESS = 0;                                    4005
   (mssys)   EXTERNAL ADDRESS = 1;                                    4006
   (mssmpsz) EXTERNAL ADDRESS = 0;                                    4007
   (msavgno) EXTERNAL ADDRESS = 1;                                    4008
   (msavrp)  EXTERNAL ADDRESS = 2;                                    4009
   (msavpf)  EXTERNAL ADDRESS = 3;                                   40010
   (msavws)  EXTERNAL ADDRESS = 4;                                   40011
```

```
% Group allocation%
   % User block flag definitions %
      (usroff)  EXTERNAL CONSTANT = 400000B;  % this user is off quota
      %                                                          41A1
      (usrnot)  EXTERNAL CONSTANT = 200000B;  % user has been notified
      of logout%                                                 41A2
      (usrexp)  EXTERNAL CONSTANT = 100000B;  % job is express %    41A3
      (usrlgd)  EXTERNAL CONSTANT =  40000B;  % this user is logged in
      %                                                          41A4
      (usrina)  EXTERNAL CONSTANT =  20000B;                     41A5
   % Core page displacement definitions %
      (grpadr)  EXTERNAL CONSTANT = 3B;                          41B1
      (expadr)  EXTERNAL CONSTANT = 103B;                        41B2
      (ptradr)  EXTERNAL CONSTANT = 203B;                        41B3
      (corlgd)  EXTERNAL CONSTANT = 304B;                        41B4
      (corlge)  EXTERNAL CONSTANT = 305B;                        41B5
      (corexr)  EXTERNAL CONSTANT = 307B;                        41B6
      (corblk)  EXTERNAL CONSTANT = 316B;                        41B7
% Create display opcodes %
   (dspyes)     EXTERNAL CONSTANT = 0;  % reformat only da's involved
   in change%                                                    42A
   (dsprfmt)    EXTERNAL CONSTANT = 1;  % reformat explicit stids %  42B
   (dspstrc)    EXTERNAL CONSTANT = 2;  % restructure only %          42C
   (dsprfst)    EXTERNAL CONSTANT = 3;  % dsprfmt + dspstrc %         42D
   (dspjpf)     EXTERNAL CONSTANT = 4;  % jump command - no editing %
                                                                 42E
   (dspno)      EXTERNAL CONSTANT = 5;  % do not reformat any da's %
                                                                 42F
% Structure codes %
   (pred)       EXTERNAL CONSTANT = 1;                        43A
   (sub)        EXTERNAL CONSTANT = 2;                        43B
   (suc)        EXTERNAL CONSTANT = 3;                        43C
   (up)         EXTERNAL CONSTANT = 4;                        43D
   (upcase)     EXTERNAL CONSTANT = 40B;                      43E
   (svmxlev)    EXTERNAL CONSTANT = 64;                       43F
      % if this is changed, change the dimensions of the arrays sqsvws
      and fvstk %
   (alphbase)   EXTERNAL CONSTANT = 26;                       43G
   (alphoff)    EXTERNAL CONSTANT = "@;                       43H
   (numbase)    EXTERNAL CONSTANT = 10;                       43I
   (numoff)     EXTERNAL CONSTANT = "0;                       43J
% Case modes %
   (lowcase)    EXTERNAL CONSTANT = 1;                        44A
   (iupcase)    EXTERNAL CONSTANT = 2;                        44B
   (supcase)    EXTERNAL CONSTANT = 4;                        44C
% String %
   (rnmdel)     EXTERNAL STRING _ " ";                        45A
% Command word values%
   (cwback) EXTERNAL CONSTANT = 59;                           46A
   (cwbranch) EXTERNAL CONSTANT =  26;                        46B
   (cwcharacter) EXTERNAL CONSTANT =  2;                      46C
   (cwcntlq) EXTERNAL CONSTANT = 51;                          46D
   (cwcontent) EXTERNAL CONSTANT = 53;                        46E
   (cwdown) EXTERNAL CONSTANT = 55;                           46F
   (cwend) EXTERNAL CONSTANT = 58;                            46G
   (cwexternal) EXTERNAL CONSTANT = 59;                       46H
```

```
   (cwextname) EXTERNAL CONSTANT = 66;                                  46I
   (cwfile) EXTERNAL CONSTANT = 51;                                     46J
   (cwfilnamed) EXTERNAL CONSTANT = 63;  %filenamed%                    46K
   (cwfirst) EXTERNAL CONSTANT = 53;                                    46L
   (cwfname) EXTERNAL CONSTANT =  12  %filename%;                       46M
   (cwfreturn) EXTERNAL CONSTANT = 67;  %filereturn%                    46N
   (cwfstcontent) EXTERNAL CONSTANT = 68;  %first content%              46O
   (cwfstname) EXTERNAL CONSTANT = 64;  %first name%                    46P
   (cwfstword) EXTERNAL CONSTANT = 70;  %first word%                    46Q
   (cwgrammar) EXTERNAL CONSTANT = 55;                                  46R
   (cwgroup) EXTERNAL CONSTANT = 27;                                    46S
   (cwhead) EXTERNAL CONSTANT = 56;                                     46T
   (cwidentlist) EXTERNAL CONSTANT =13;                                 46U
   (cwinvisible) EXTERNAL CONSTANT = 11;                                46V
   (cwlink) EXTERNAL CONSTANT =  30;                                    46W
   (cwlocation) EXTERNAL CONSTANT =  29;                                46X
   (cwname) EXTERNAL CONSTANT =  32;                                    46Y
   (cwnewfilename) EXTERNAL CONSTANT = 6;                               46Z
   (cwnext) EXTERNAL CONSTANT = 61;                                    46A@
   (cwnumber) EXTERNAL CONSTANT =  8;                                   46AA
   (cwnxtcontent) EXTERNAL CONSTANT = 69;  %next content%               46AB
   (cwnxtname) EXTERNAL CONSTANT = 65;  %next name%                     46AC
   (cwnxtword) EXTERNAL CONSTANT = 71;  %next word%                     46AD
   (cwoff) EXTERNAL CONSTANT = 63;                                      46AE
   (cwoldfilename) EXTERNAL CONSTANT =  7;                              46AF
   (cwon) EXTERNAL CONSTANT = 62;                                       46AG
   (cworigin) EXTERNAL CONSTANT = 60;                                   46AH
   (cwpassword) EXTERNAL CONSTANT = 10;                                 46AI
   (cwplex) EXTERNAL CONSTANT =  28;                                    46AJ
   (cwpredecessor) EXTERNAL CONSTANT = 53;                              46AK
   (cwreturn) EXTERNAL CONSTANT = 62;                                   46AL
   (cwstatement) EXTERNAL CONSTANT =  29;                               46AM
   (cwsuccessor) EXTERNAL CONSTANT = 52;                                46AN
   (cwtail) EXTERNAL CONSTANT = 57;                                     46AO
   (cwtext) EXTERNAL CONSTANT =  1;                                     46AP
   (cwup) EXTERNAL CONSTANT = 54;                                       46AQ
   (cwviewspecs) EXTERNAL CONSTANT = 57;                                46AR
   (cwvisible) EXTERNAL CONSTANT =  4;                                  46AS
   (cwwindow) EXTERNAL CONSTANT =  33;                                  46AT
   (cword) EXTERNAL CONSTANT =  3;                                      46AU
% For use with syntax generating commands %
   (nmalt)      EXTERNAL CONSTANT = 5;                                   47A
      % if a rule has more than nmalt paths, then the rule will not be
      expanded and the name of the rule will appear %
% Miscellaneous (previously file) globals %
   % Constants used in PSEDT1 for clist parameters %
      (nofile)  EXTERNAL CONSTANT = 0;                                  48A1
      (noct)    EXTERNAL CONSTANT = 0;                                  48A2
      (ctcsp)   EXTERNAL CONSTANT = 1;                                  48A3
      (ctfrz)   EXTERNAL CONSTANT = 2;                                  48A4
      (ctcpfz)  EXTERNAL CONSTANT = 3;                                  48A5
      (ctmkr)   EXTERNAL CONSTANT = 8;                                  48A6
      (ctcmk)   EXTERNAL CONSTANT = 9;                                  48A7
      (ctlcfm)  EXTERNAL CONSTANT = 11;                                 48A8
      (ctlcfm)  EXTERNAL CONSTANT = 15;                                 48A9
FINISH
```

(acchecked)	<nine, bconst, 0204>	EXT	7I1
(acor2)	<nine, bconst, 0444>	EXT CONSTANT =43B	33F1A4
(acor4)	<nine, bconst, 0445>	EXT CONSTANT =45B	33F1A5
(afgtyp)	<nine, bconst, 0380>	EXT CONSTANT =4	29D
(alphbase)	<nine, bconst, 0532>	EXT CONSTANT =26	43G
(alphoff)	<nine, bconst, 0767>	EXT CONSTANT ="@	43H
(append)	<nine, bconst, 0664>	EXT CONSTANT =3	7L4
(apsda)	<nine, bconst, 0401>	EXT CONSTANT =17B	31A8
(archost)	<nine, bconst, 0501>	EXT CONSTANT =150	37A
(arcistr)	<nine, bconst, 0990>	EXT STRING	37N
(arcoprtty)	<nine, bconst, 0562>	EXT CONSTANT =27B	10B1
(ascalt)	<nine, bconst, 0142>	EXT ADDRESS =33B	5T
(ascbst)	<nine, bconst, 0124>	EXT ADDRESS =21B	5A
(asctsw)	<nine, bconst, 0125>	EXT ADDRESS =36B	5B
(assfil)	<nine, bconst, 0637>	EXT CONSTANT =3	7G5
(astr)	<nine, bconst, 0705>	EXT STRING	33H7
(bbnbhost)	<nine, bconst, 0839>	EXT CONSTANT =49	37L
(bcstr)	<nine, bconst, 0708>	EXT STRING	33H5
(begmsg)	<nine, bconst, 0408>	EXT CONSTANT =33B	31C
(bfile)	<nine, bconst, 0590>	EXT CONSTANT =4	7M1
(bfree)	<nine, bconst, 032>	EXT ADDRESS =4000008	3D
(bhi)	<nine, bconst, 0490>	EXT CONSTANT =1	34C
(bintyp)	<nine, bconst, 0201>	EXT CONSTANT =2	7G12
(blkoffset)	<nine, bconst, 0491>	EXT CONSTANT =0	34D
(blkprd)	<nine, bconst, 0489>	EXT CONSTANT =2	34B
(blksiz)	<nine, bconst, 0163>	EXT CONSTANT =512	7A14
(blksuc)	<nine, bconst, 0488>	EXT CONSTANT =1	34A
(braname)	<nine, bconst, 0363>	EXT CONSTANT =9	25H
(brnchv)	<nine, bconst, 0336>	EXT CONSTANT =2	22B
(bstr)	<nine, bconst, 0701>	EXT STRING	33H8
(caext)	<nine, bconst, 0849>	EXT STRING	7J9
(castr)	<nine, bconst, 0474>	EXT STRING	33H1
(cawait)	<nine, bconst, 0421>	EXT CONSTANT =1	32D2
(cawtnomes)	<nine, bconst, 0991>	EXT CONSTANT =2	32D3
(ccfda)	<nine, bconst, 0404>	EXT CONSTANT =22B	31A11
(ccnda)	<nine, bconst, 0403>	EXT CONSTANT =21B	31A10
(cdotstr)	<nine, bconst, 0710>	EXT STRING	33H2
(cdstr)	<nine, bconst, 0707>	EXT STRING	33H4
(ce)	<nine, bconst, 0249>	EXT CONSTANT =7	15E
(cgrext)	<nine, bconst, 0881>	EXT STRING	7J7
(charv)	<nine, bconst, 0342>	EXT CONSTANT =8	22H
(chkpcf)	<nine, bconst, 0202>	EXT CONSTANT =TRUE	7G17
(chrtyp)	<nine, bconst, 0631>	EXT CONSTANT =3	7G13
(chtyp)	<nine, bconst, 0168>	EXT CONSTANT =3	7B1D
(ckpag)	<nine, bconst, 040>	EXT ADDRESS =553000B	3F
(cmilink)	<nine, bconst, 0855>	EXT STRING	18A3
(cmptyp)	<nine, bconst, 0216>	EXT CONSTANT =2	8A2
(cnch2)	<nine, bconst, 0443>	EXT CONSTANT =41B	33F1A3
(comdir)	<nine, bconst, 0194>	EXT STRING	7F2
(comtyp)	<nine, bconst, 0630>	EXT CONSTANT =5	7G14
(contls)	<nine, bconst, 0360>	EXT CONSTANT =6	25E
(contnt)	<nine, bconst, 0358>	EXT CONSTANT =3	25C
(copyflag)	<nine, bconst, 0604>	EXT CONSTANT =1	9B1
(corblk)	<nine, bconst, 0753>	EXT CONSTANT =316B	41B7
(corexr)	<nine, bconst, 0754>	EXT CONSTANT =307B	41B6
(corlgd)	<nine, bconst, 0756>	EXT CONSTANT =304B	41B4

(corlge)	<nine, bconst, 0755>	EXT CONSTANT =305B	41B5
(cortn)	<nine, bconst, 0579>	EXT CONSTANT =1	32B1
(cpcldelim)	<nine, bconst, 0823>	EXT	38B
(cpcrdelim)	<nine, bconst, 0824>	EXT	38C
(cprcdname)	<nine, bconst, 0853>	EXT STRING	18A1
(crpgad)	<nine, bconst, 0178>	EXT	7E1
(crstr)	<nine, bconst, 0713>	EXT STRING	33H33
(cs)	<nine, bconst, 0248>	EXT CONSTANT =5	15D
(cstr)	<nine, bconst, 0704>	EXT STRING	33H9
(ctctm)	<nine, bconst, 0615>	EXT CONSTANT =11	48A8
(ctcmk)	<nine, bconst, 0614>	EXT CONSTANT =9	48A7
(ctcptz)	<nine, bconst, 0612>	EXT CONSTANT =3	48A5
(ctcsp)	<nine, bconst, 0610>	EXT CONSTANT =1	48A3
(cttrz)	<nine, bconst, 0611>	EXT CONSTANT =2	48A4
(ctla)	<nine, bconst, 0126>	EXT ADDRESS =1	5C
(ctlctm)	<nine, bconst, 0616>	EXT CONSTANT =15	48A9
(ctld)	<nine, bconst, 0862>	EXT ADDRESS =4	5D
(ctle)	<nine, bconst, 0127>	EXT ADDRESS =5	5E
(ctlext)	<nine, bconst, 0208>	EXT STRING	7J4
(ctlf)	<nine, bconst, 0128>	EXT ADDRESS =6	5F
(ctlg)	<nine, bconst, 0129>	EXT ADDRESS =7	5G
(ctln)	<nine, bconst, 0130>	EXT ADDRESS =16B	5H
(ctlo)	<nine, bconst, 0131>	EXT ADDRESS =17B	5I
(ctlp)	<nine, bconst, 0132>	EXT ADDRESS =20B	5J
(ctlr)	<nine, bconst, 0133>	EXT ADDRESS =22B	5K
(ctls)	<nine, bconst, 0134>	EXT ADDRESS =23B	5L
(ctlt)	<nine, bconst, 0135>	EXT ADDRESS =24B	5M
(ctlu)	<nine, bconst, 0136>	EXT ADDRESS =25B	5N
(ctlv)	<nine, bconst, 0137>	EXT ADDRESS =26B	5O
(ctlw)	<nine, bconst, 0138>	EXT ADDRESS =27B	5P
(ctlx)	<nine, bconst, 0139>	EXT ADDRESS =30B	5Q
(ctly)	<nine, bconst, 0140>	EXT ADDRESS =31B	5R
(ctlz)	<nine, bconst, 0141>	EXT ADDRESS =32B	5S
(ctmkr)	<nine, bconst, 0613>	EXT CONSTANT =8	48A6
(cwacc)	<nine, bconst, 0559>	EXT CONSTANT =400B6	10D5
(cwback)	<nine, bconst, 0921>	EXT CONSTANT =59	46A
(cwbranch)	<nine, bconst, 0911>	EXT CONSTANT =26	46B
(cwcharacter)	<nine, bconst, 0892>	EXT CONSTANT =2	46C
(cwcntlg)	<nine, bconst, 0923>	EXT CONSTANT =51	46D
(cwcontent)	<nine, bconst, 0922>	EXT CONSTANT =53	46E
(cwdown)	<nine, bconst, 0924>	EXT CONSTANT =55	46F
(cweditent)	<nine, bconst, 0960>	EXT CONSTANT =0	23E
(cwend)	<nine, bconst, 0925>	EXT CONSTANT =58	46G
(cwexternal)	<nine, bconst, 0926>	EXT CONSTANT =59	46H
(cwextname)	<nine, bconst, 0927>	EXT CONSTANT =66	46I
(cwfile)	<nine, bconst, 0928>	EXT CONSTANT =51	46J
(cwfilnamed)	<nine, bconst, 0961>	EXT CONSTANT =63	46K
(cwfirst)	<nine, bconst, 0931>	EXT CONSTANT =53	46L
(cwfname)	<nine, bconst, 0890>	EXT CONSTANT =12	%filename%
(cwfreturn)	<nine, bconst, 0930>	EXT CONSTANT =67	46N
(cwfstcontent)	<nine, bconst, 0932>	EXT CONSTANT =68	46O
(cwfstname)	<nine, bconst, 0933>	EXT CONSTANT =64	46P
(cwfstword)	<nine, bconst, 0934>	EXT CONSTANT =70	46Q
(cwgrammar)	<nine, bconst, 0935>	EXT CONSTANT =55	46R
(cwgroup)	<nine, bconst, 0915>	EXT CONSTANT =27	46S
(cwhead)	<nine, bconst, 0936>	EXT CONSTANT =56	46T

```
(cwidentlist)  <nine, bconst, 0937>     EXT CONSTANT =13      46U
(cwinvisible)  <nine, bconst, 0899>     EXT CONSTANT =11      46V
(cwlink)       <nine, bconst, 0900>     EXT CONSTANT =30      46W
(cwlocation)   <nine, bconst, 0909>     EXT CONSTANT =29      46X
(cwname)       <nine, bconst, 0917>     EXT CONSTANT =32      46Y
(cwnewfilename) <nine, bconst, 0886>    EXT CONSTANT =6       46Z
(cwnext)       <nine, bconst, 0938>     EXT CONSTANT =61      46A@
(cwnumber)     <nine, bconst, 0902>     EXT CONSTANT =8       46AA
(cwnxtcontent) <nine, bconst, 0939>     EXT CONSTANT =69      46AB
(cwnxtname)    <nine, bconst, 0940>     EXT CONSTANT =65      46AC
(cwnxtword)    <nine, bconst, 0941>     EXT CONSTANT =71      46AD
(cwoff)        <nine, bconst, 0942>     EXT CONSTANT =63      46AE
(cwoldfilename) <nine, bconst, 0888>    EXT CONSTANT =7       46AF
(cwon)         <nine, bconst, 0943>     EXT CONSTANT =62      46AG
(cworigin)     <nine, bconst, 0944>     EXT CONSTANT =60      46AH
(cwpassword)   <nine, bconst, 0906>     EXT CONSTANT =10      46AI
(cwplex)       <nine, bconst, 0913>     EXT CONSTANT =28      46AJ
(cwpredecessor) <nine, bconst, 0945>    EXT CONSTANT =53      46AK
(cwreturn)     <nine, bconst, 0946>     EXT CONSTANT =62      46AL
(cwstatement)  <nine, bconst, 0907>     EXT CONSTANT =29      46AM
(cwstring)     <nine, bconst, 0958>     EXT CONSTANT =100     23C
(cwstructure)  <nine, bconst, 0959>     EXT CONSTANT =101     23D
(cwsuccessor)  <nine, bconst, 0947>     EXT CONSTANT =52      46AN
(cwtail)       <nine, bconst, 0948>     EXT CONSTANT =57      46AO
(cwtext)       <nine, bconst, 0896>     EXT CONSTANT =1       46AP
(cwtype)       <nine, bconst, 0416>     EXT CONSTANT =1       32A1
(cwup)         <nine, bconst, 0949>     EXT CONSTANT =54      46AQ
(cwviewspecs)  <nine, bconst, 0950>     EXT CONSTANT =57      46AR
(cwvisible)    <nine, bconst, 0898>     EXT CONSTANT =4       46AS
(cwwindow)     <nine, bconst, 0918>     EXT CONSTANT =33      46AT
(cwword)       <nine, bconst, 0894>     EXT CONSTANT =3       46AU
(de)           <nine, bconst, 0257>     EXT CONSTANT =23      15M
(deltflag)     <nine, bconst, 0674>     EXT CONSTANT =4       9B4
(difvrs)       <nine, bconst, 0626>     EXT CONSTANT =FALSE   7C20
(dhtyp)        <nine, bconst, 0169>     EXT CONSTANT =4       7B1E
(dirstname)    <nine, bconst, 0828>     EXT STRING            38E1
(dntyp)        <nine, bconst, 0566>     EXT CONSTANT =0       10A1
(down)         <nine, bconst, 0685>     EXT CONSTANT =-1      21D
(downstr)      <nine, bconst, 0331>     EXT STRING            21A
(dpybks)       <nine, bconst, 043>      EXT CONSTANT =53000B  4A
(dpycoc)       <nine, bconst, 0369>     EXT CONSTANT =1040125373 26B
(ds)           <nine, bconst, 0256>     EXT CONSTANT =21      15L
(dspallf)      <nine, bconst, 0694>     EXT CONSTANT =6       33D
(dspbke)       <nine, bconst, 030>      EXT ADDRESS =757777B  3C
(dspbik)       <nine, bconst, 0871>     EXT ADDRESS =705000B  3B
(dspjpf)       <nine, bconst, 0761>     EXT CONSTANT =4       42E
(dspno)        <nine, bconst, 0760>     EXT CONSTANT =5       42F
(dsprfmt)      <nine, bconst, 0764>     EXT CONSTANT =1       42B
(dsprfst)      <nine, bconst, 0762>     EXT CONSTANT =3       42D
(dspstrc)      <nine, bconst, 0763>     EXT CONSTANT =2       42C
(dspyes)       <nine, bconst, 0759>     EXT CONSTANT =0       42A
(dtbbas)       <nine, bconst, 0153>     EXT CONSTANT =101     7A4
(dtbm)         <nine, bconst, 0154>     EXT CONSTANT =370     7A5
(eacc)         <nine, bconst, 0557>     EXT CONSTANT =20000B6 10D3
(edit)         <nine, bconst, 0145>     EXT ADDRESS =2        6B
(eltee)        <nine, bconst, 0620>     EXT ADDRESS =554000B  3E
```

(emreset)	<nine, bconst, 0741>	EXT ADDRESS =3	39C
(emrtnfalse)	<nine, bconst, 0508>	EXT ADDRESS =1	39A
(emsignal)	<nine, bconst, 0740>	EXT ADDRESS =2	39B
(enistr)	<nine, bconst, 0719>	EXT STRING	33H34
(escstr)	<nine, bconst, 0706>	EXT STRING	33H6
(estr)	<nine, bconst, 0703>	EXT STRING	33H10
(exeext)	<nine, bconst, 0861>	EXT STRING	7J3
(expadr)	<nine, bconst, 0757>	EXT CONSTANT =103B	41B2
(extname)	<nine, bconst, 0364>	EXT CONSTANT =10	25I
(fbhdl)	<nine, bconst, 0156>	EXT CONSTANT =2	7A7
(fcor2)	<nine, bconst, 0441>	EXT CONSTANT =42B	33F1A1
(fcor4)	<nine, bconst, 0442>	EXT CONSTANT =44B	33F1A2
(fe)	<nine, bconst, 0255>	EXT CONSTANT =19	15K
(filev)	<nine, bconst, 0340>	EXT CONSTANT =6	22F
(flgtname)	<nine, bconst, 0829>	EXT STRING	38E2
(fnormal)	<nine, bconst, 0692>	EXT CONSTANT =0	33B
(fontmax)	<nine, bconst, 0437>	EXT CONSTANT =512	33A
(forwdv)	<nine, bconst, 0350>	EXT CONSTANT =16	22P
(frrmax)	<nine, bconst, 0469>	EXT CONSTANT =25	33G1
(fs)	<nine, bconst, 0254>	EXT CONSTANT =17	15J
(fstr)	<nine, bconst, 0702>	EXT STRING	33H11
(fulldisplay)	<nine, bconst, 0691>	EXT CONSTANT =1	32F2
(fvrdchar)	<nine, bconst, 0820>	EXT	38A
(groupv)	<nine, bconst, 0337>	EXT CONSTANT =3	22C
(grpadr)	<nine, bconst, 0527>	EXT CONSTANT =3B	41B1
(grptname)	<nine, bconst, 0830>	EXT STRING	38E3
(grptyp)	<nine, bconst, 0378>	EXT CONSTANT =2	29B
(gstr)	<nine, bconst, 0700>	EXT STRING	33H12
(gtttyp)	<nine, bconst, 0166>	EXT CONSTANT =1	7B1B
(gtjcpc)	<nine, bconst, 0585>	EXT CONSTANT =2B3	7K10D
(gtjidl)	<nine, bconst, 0658>	EXT CONSTANT =1B9	7K8
(gtjigdel)	<nine, bconst, 0667>	EXT CONSTANT =1B3	7K10C
(gtjnfo)	<nine, bconst, 0657>	EXT CONSTANT =2B11	7K9
(gtjoif)	<nine, bconst, 0653>	EXT CONSTANT =1B11	7K2
(gtjoisf)	<nine, bconst, 0660>	EXT CONSTANT =1B11	7K5
(gtjoof)	<nine, bconst, 0210>	EXT CONSTANT =4B11	7K1
(gtjoost)	<nine, bconst, 0655>	EXT CONSTANT =4B11	7K4
(gtjprf)	<nine, bconst, 0654>	EXT CONSTANT =1B11	7K3
(gtjprv)	<nine, bconst, 0659>	EXT CONSTANT =2B9	7K7
(gtjred)	<nine, bconst, 0666>	EXT CONSTANT =1B5	7K10B
(gtjstr)	<nine, bconst, 0661>	EXT CONSTANT =1B8	7K6
(gtjwrt)	<nine, bconst, 0584>	EXT CONSTANT =4B5	7K10A
(hcopyv)	<nine, bconst, 0341>	EXT CONSTANT =7	22G
(hdtyp)	<nine, bconst, 0158>	EXT CONSTANT =0	7A9
(he)	<nine, bconst, 0251>	EXT CONSTANT =11	15G
(heurtil)	<nine, bconst, 0636>	EXT CONSTANT =88	7G6
(horizontal)	<nine, bconst, 0597>	EXT CONSTANT =0	33E1
(hs)	<nine, bconst, 0250>	EXT CONSTANT =9	15F
(hstr)	<nine, bconst, 0699>	EXT STRING	33H13
(indtyp)	<nine, bconst, 0377>	EXT CONSTANT =1	29A
(inremsiteflag)	<nine, bconst, 0505>	EXT CONSTANT =TRUE	37S
(invisv)	<nine, bconst, 0346>	EXT CONSTANT =12	22L
(irep)	<nine, bconst, 0446>	EXT CONSTANT =46B	33F1A6
(isichost)	<nine, bconst, 0834>	EXT CONSTANT =150	37F
(isidhost)	<nine, bconst, 0833>	EXT CONSTANT =214	37G
(isiehost)	<nine, bconst, 0836>	EXT CONSTANT =116	37H

(isirlhost)	<nine, bconst, 0846>	EXT CONSTANT =33	37I
(iupcase)	<nine, bconst, 0776>	EXT CONSTANT =2	44B
(jfnnamonly)	<nine, bconst, 0669>	EXT CONSTANT =1B9	7K13
(jfnstf)	<nine, bconst, 0587>	EXT CONSTANT =111100400018	
(jnktyp)	<nine, bconst, 0161>	EXT CONSTANT =3	7A12
(jnlnam)	<nine, bconst, 0688>	EXT STRING	28D
(jnlpsw)	<nine, bconst, 0496>	EXT STRING	36A1
(jnltime)	<nine, bconst, 0498>	EXT	36B1
(jump)	<nine, bconst, 0146>	EXT ADDRESS =3	6C
(justtil)	<nine, bconst, 0635>	EXT CONSTANT =89	7G7
(kstr)	<nine, bconst, 0697>	EXT STRING	33H14
(l10link)	<nine, bconst, 0854>	EXT STRING	18A2
(le)	<nine, bconst, 0247>	EXT CONSTANT =3	15C
(levdown)	<nine, bconst, 0242>	EXT CONSTANT =-1	14C
(levsuc)	<nine, bconst, 0240>	EXT CONSTANT =0	14A
(levup)	<nine, bconst, 0241>	EXT CONSTANT =1	14B
(lfdstr)	<nine, bconst, 0709>	EXT STRING	33H3
(lfn)	<nine, bconst, 0245>	EXT CONSTANT =0	15A
(lgngrps)	<nine, bconst, 0645>	EXT CONSTANT =0	7I3
(linkv)	<nine, bconst, 0347>	EXT CONSTANT =13	22M
(litv)	<nine, bconst, 0339>	EXT CONSTANT =5	22E
(lnkdsl)	<nine, bconst, 0260>	EXT CONSTANT =29	15P
(lowcase)	<nine, bconst, 0534>	EXT CONSTANT =1	44A
(lpbug)	<nine, bconst, 0455>	EXT CONSTANT =46B	33F1J
(lpcline)	<nine, bconst, 0452>	EXT CONSTANT =43B	33F1G
(lpcoord)	<nine, bconst, 0463>	EXT CONSTANT =61B	33F1R
(lpcscreen)	<nine, bconst, 0457>	EXT CONSTANT =50B	33F1L
(lpdline)	<nine, bconst, 0453>	EXT CONSTANT =44B	33F1H
(lpendstandout)	<nine, bconst, 0461>	EXT CONSTANT =57B	33F1P
(lpesc)	<nine, bconst, 0447>	EXT CONSTANT =33B	33F1B
(lpiniine)	<nine, bconst, 0454>	EXT CONSTANT =45B	33F1I
(lpintter)	<nine, bconst, 0459>	EXT CONSTANT =55B	33F1N
(lpnocoor)	<nine, bconst, 0462>	EXT CONSTANT =60B	33F1Q
(lppbug)	<nine, bconst, 0456>	EXT CONSTANT =47B	33F1K
(lpposition)	<nine, bconst, 0449>	EXT CONSTANT =40B	33F1D
(lpresc)	<nine, bconst, 0448>	EXT CONSTANT =34B	33F1C
(lpreset)	<nine, bconst, 0458>	EXT CONSTANT =51B	33F1M
(lpstandout)	<nine, bconst, 0460>	EXT CONSTANT =56B	33F1O
(lptptclose)	<nine, bconst, 0465>	EXT CONSTANT =54B	33F1T
(lptptopen)	<nine, bconst, 0464>	EXT CONSTANT =53B	33F1S
(lptptsr)	<nine, bconst, 0466>	EXT CONSTANT =52B	33F1U
(lptrack)	<nine, bconst, 0451>	EXT CONSTANT =42B	33F1F
(lptty)	<nine, bconst, 0450>	EXT CONSTANT =41B	33F1E
(lpttype)	<nine, bconst, 0629>	EXT CONSTANT =6	7G15
(ls)	<nine, bconst, 0246>	EXT CONSTANT =1	15B
(lseltyp)	<nine, bconst, 0956>	EXT CONSTANT =1	23A
(lstr)	<nine, bconst, 0698>	EXT STRING	33H15
(lwtyp)	<nine, bconst, 0167>	EXT CONSTANT =2	7B1C
(mactil)	<nine, bconst, 0638>	EXT CONSTANT =2	7G4
(maxbsize)	<nine, bconst, 0319>	EXT CONSTANT =44	17B
(maxdpschar)	<nine, bconst, 0582>	EXT CONSTANT =1750	32C1
(minfpages)	<nine, bconst, 0873>	EXT CONSTANT =20	17C
(minwb)	<nine, bconst, 0599>	EXT CONSTANT =2	33E4
(minww)	<nine, bconst, 0598>	EXT CONSTANT =20	33E3
(moveflag)	<nine, bconst, 0676>	EXT CONSTANT =2	9B2
(msavpf)	<nine, bconst, 0779>	EXT ADDRESS =3	40010

(msavgno)	<nine, bconst, 0781>	EXT ADDRESS =1	4008
(msavrp)	<nine, bconst, 0780>	EXT ADDRESS =2	4009
(msavws)	<nine, bconst, 0778>	EXT ADDRESS =4	40011
(msbavp)	<nine, bconst, 0750>	EXT CONSTANT =30	40J
(msbk11)	<nine, bconst, 0513>	EXT CONSTANT =3	40G
(msbk21)	<nine, bconst, 0752>	EXT CONSTANT =4	40H
(msbuf1)	<nine, bconst, 0749>	EXT CONSTANT =512	40L
(mschrt)	<nine, bconst, 0744>	EXT CONSTANT =2	40C
(mscount)	<nine, bconst, 0785>	EXT ADDRESS =2	4003
(mscurt)	<nine, bconst, 0783>	EXT ADDRESS =0	4005
(msdsys)	<nine, bconst, 0748>	EXT CONSTANT =0	40M
(mstinish)	<nine, bconst, 0742>	EXT CONSTANT =0	40F
(mshdr1)	<nine, bconst, 0751>	EXT CONSTANT =2	40I
(mspmtot)	<nine, bconst, 0784>	EXT ADDRESS =3	4004
(msproc)	<nine, bconst, 0516>	EXT ADDRESS =0	4001
(msreal)	<nine, bconst, 0786>	EXT ADDRESS =1	4002
(msrecl)	<nine, bconst, 0514>	EXT CONSTANT =35	40K
(mssmpsz)	<nine, bconst, 0517>	EXT ADDRESS =0	4007
(msstart)	<nine, bconst, 0743>	EXT CONSTANT =1	40E
(msstre)	<nine, bconst, 0745>	EXT CONSTANT =1	40B
(mssys)	<nine, bconst, 0782>	EXT ADDRESS =1	4006
(mstr)	<nine, bconst, 0696>	EXT STRING	33H16
(mstsys)	<nine, bconst, 0747>	EXT CONSTANT =1	40N
(mstxte)	<nine, bconst, 0511>	EXT CONSTANT =0	40A
(mswsiw)	<nine, bconst, 0746>	EXT CONSTANT =3	40D
(n40fil)	<nine, bconst, 0199>	EXT CONSTANT =0	7G2
(nametyp)	<nine, bconst, 0356>	EXT CONSTANT =1	25A
(netype)	<nine, bconst, 0628>	EXT CONSTANT =7	7G16
(newname)	<nine, bconst, 0673>	EXT CONSTANT =4	9A4
(newversion)	<nine, bconst, 0602>	EXT CONSTANT =1	9A1
(niltyp)	<nine, bconst, 0162>	EXT CONSTANT =4	7A13
(nlsext)	<nine, bconst, 0206>	EXT STRING	7J1
(nmait)	<nine, bconst, 0538>	EXT CONSTANT =5	47A
(nocawait)	<nine, bconst, 0422>	EXT CONSTANT =0	32D1
(noct)	<nine, bconst, 0609>	EXT CONSTANT =0	48A2
(notile)	<nine, bconst, 0608>	EXT CONSTANT =0	48A1
(npcodes)	<nine, bconst, 0479>	EXT	33J1
(npsize)	<nine, bconst, 0485>	EXT CONSTANT =36	33L1
(npstrings)	<nine, bconst, 0483>	EXT	33K1
(nsahost)	<nine, bconst, 0798>	EXT CONSTANT =65	37C
(nsaoprtty)	<nine, bconst, 0864>	EXT CONSTANT =0B	10B3
(nsastr)	<nine, bconst, 0863>	EXT STRING	37Q
(nstr)	<nine, bconst, 0725>	EXT STRING	33H17
(nttyp)	<nine, bconst, 0568>	EXT CONSTANT =2	10A3
(nuistr)	<nine, bconst, 0720>	EXT STRING	33H36
(numbase)	<nine, bconst, 0766>	EXT CONSTANT =10	43I
(numbrv)	<nine, bconst, 0344>	EXT CONSTANT =10	22J
(numoff)	<nine, bconst, 0765>	EXT CONSTANT =0	43J
(nvtk)	<nine, bconst, 0198>	EXT CONSTANT =2	7G1
(nwada)	<nine, bconst, 0400>	EXT CONSTANT =16B	31A7
(nwheecinits)	<nine, bconst, 0550>	EXT	10C1
(nwstrda)	<nine, bconst, 0402>	EXT CONSTANT =20B	31A9
(nxtname)	<nine, bconst, 0361>	EXT CONSTANT =7	25F
(odtyp)	<nine, bconst, 0217>	EXT CONSTANT =3	8A3
(office1)	<nine, bconst, 0953>	EXT CONSTANT =43	37E
(oldpgsz)	<nine, bconst, 0494>	EXT CONSTANT =0	35A

(oldversion)	<nine, bconst, 0671>	EXT CONSTANT =2 9A2
(oldvrsn)	<nine, bconst, 0640>	EXT CONSTANT =-4 7G19
(opcmdv)	<nine, bconst, 0386>	EXT CONSTANT =3 30A2
(opmctl)	<nine, bconst, 0633>	EXT CONSTANT =3 7G10
(opmitl)	<nine, bconst, 0632>	EXT CONSTANT =4 7G11
(opname)	<nine, bconst, 0372>	EXT STRING 28A
(opnmod)	<nine, bconst, 0882>	EXT CONSTANT =1 7L7
(opprdv)	<nine, bconst, 0387>	EXT CONSTANT =4 30A3
(opqptl)	<nine, bconst, 0634>	EXT CONSTANT =2 7G9
(oprmdv)	<nine, bconst, 0391>	EXT CONSTANT =10 30A6
(opsqtl)	<nine, bconst, 0200>	EXT CONSTANT =1 7G8
(optydv)	<nine, bconst, 0385>	EXT CONSTANT =2 30A1
(optytl)	<nine, bconst, 0987>	EXT CONSTANT =11 30A7
(opxpdv)	<nine, bconst, 0389>	EXT CONSTANT =6 30A4
(opxtdv)	<nine, bconst, 0390>	EXT CONSTANT =7 30A5
(orgstid)	<nine, bconst, 0177>	EXT CONSTANT =2 7D1
(orgtyp)	<nine, bconst, 0379>	EXT CONSTANT =3 29C
(origft)	<nine, bconst, 0627>	EXT CONSTANT =FALSE 7G18
(ostr)	<nine, bconst, 0724>	EXT STRING 33H18
(outremsiteflag)	<nine, bconst, 0506>	EXT CONSTANT =TRUE 37T
(pconlen)	<nine, bconst, 0236>	EXT CONSTANT =1 13C1
(pkeylen)	<nine, bconst, 0237>	EXT CONSTANT =1 13D1
(plexv)	<nine, bconst, 0338>	EXT CONSTANT =4 22D
(plvlen)	<nine, bconst, 0235>	EXT CONSTANT =1 13B1
(prcadr)	<nine, bconst, 0218>	EXT CONSTANT =400010B 8B1
(pred)	<nine, bconst, 0531>	EXT CONSTANT =1 43A
(procext)	<nine, bconst, 0850>	EXT STRING 7J12
(proptab)	<nine, bconst, 0171>	EXT 7C1
(psprivate)	<nine, bconst, 0641>	EXT ADDRESS =2 7H2
(pspublic)	<nine, bconst, 0203>	EXT ADDRESS =1 7H1
(pstr)	<nine, bconst, 0726>	EXT STRING 33H19
(ptradr)	<nine, bconst, 0758>	EXT CONSTANT =203B 41B3
(pvslen)	<nine, bconst, 0234>	EXT CONSTANT =3 13A1
(qstr)	<nine, bconst, 0723>	EXT STRING 33H20
(racc)	<nine, bconst, 0555>	EXT CONSTANT =100000B6 10D1
(random)	<nine, bconst, 0353>	EXT CONSTANT =4 24A
(read)	<nine, bconst, 0656>	EXT CONSTANT =1 7L2
(readwrite)	<nine, bconst, 0663>	EXT CONSTANT =2 7L3
(regrtn)	<nine, bconst, 0580>	EXT CONSTANT =0 32B2
(relext)	<nine, bconst, 0195>	EXT STRING 7J5
(remada)	<nine, bconst, 0394>	EXT CONSTANT =1 31A1
(remdda)	<nine, bconst, 0395>	EXT CONSTANT =2 31A2
(remfudge)	<nine, bconst, 0407>	EXT CONSTANT =40B 31B
(reminit)	<nine, bconst, 0405>	EXT CONSTANT =33B 31A12
(remrdda)	<nine, bconst, 0410>	EXT CONSTANT =7 31E
(remscm)	<nine, bconst, 0399>	EXT CONSTANT =15B 31A6
(remscsr)	<nine, bconst, 0397>	EXT CONSTANT =5 31A4
(remsdda)	<nine, bconst, 0409>	EXT CONSTANT =6 31D
(remssda)	<nine, bconst, 0398>	EXT CONSTANT =10B 31A5
(remstr)	<nine, bconst, 0396>	EXT CONSTANT =4 31A3
(remtsf)	<nine, bconst, 0412>	EXT CONSTANT =13B 31G
(remtsn)	<nine, bconst, 0411>	EXT CONSTANT =12B 31F
(rfchost)	<nine, bconst, 0502>	EXT CONSTANT =66 37D
(rfpmax)	<nine, bconst, 0150>	EXT CONSTANT =100B 7A1
(ringl)	<nine, bconst, 0155>	EXT CONSTANT =5 7A6
(rngbas)	<nine, bconst, 0151>	EXT CONSTANT =6 7A2

(rngm)	<nine, bconst, 0152>	EXT CONSTANT =95	7A3
(rngtyp)	<nine, bconst, 0160>	EXT CONSTANT =2	7A11
(rnmdel)	<nine, bconst, 0535>	EXT STRING	45A
(rstr)	<nine, bconst, 0721>	EXT STRING	33H21
(rthawed)	<nine, bconst, 0662>	EXT CONSTANT =5	7L5
(rtmax)	<nine, bconst, 0693>	EXT CONSTANT =60	33C
(rwthawed)	<nine, bconst, 0665>	EXT CONSTANT =4	7L6
(savext)	<nine, bconst, 0207>	EXT STRING	7J2
(sdbtyp)	<nine, bconst, 0159>	EXT CONSTANT =1	7A10
(seqname)	<nine, bconst, 0362>	EXT CONSTANT =8	25G
(sgext)	<nine, bconst, 0879>	EXT STRING	7J11
(sgtjtf)	<nine, bconst, 0668>	EXT CONSTANT =1B6	7K11
(sid)	<nine, bconst, 0359>	EXT CONSTANT =4	25D
(sjaccess)	<nine, bconst, 0973>	EXT STRING	19A
(sjaction)	<nine, bconst, 0965>	EXT STRING	19B
(sjauthor)	<nine, bconst, 0978>	EXT STRING	19C
(sjbranch)	<nine, bconst, 0969>	EXT STRING	19D
(sjcomments)	<nine, bconst, 0980>	EXT STRING	19E
(sjfile)	<nine, bconst, 0977>	EXT STRING	19F
(sjforward)	<nine, bconst, 0983>	EXT STRING	19G
(sjgroup)	<nine, bconst, 0967>	EXT STRING	19H
(sjhandle)	<nine, bconst, 0981>	EXT STRING	19I
(sjhardcopy)	<nine, bconst, 0986>	EXT STRING	19J
(sjinto)	<nine, bconst, 0984>	EXT STRING	19K
(sjkeyword)	<nine, bconst, 0979>	EXT STRING	19L
(sjlink)	<nine, bconst, 0972>	EXT STRING	19M
(sjmessage)	<nine, bconst, 0974>	EXT STRING	19N
(sjnumber)	<nine, bconst, 0968>	EXT STRING	19O
(sjobsolete)	<nine, bconst, 0970>	EXT STRING	19P
(sjplex)	<nine, bconst, 0964>	EXT STRING	19Q
(sjrecord)	<nine, bconst, 0971>	EXT STRING	19R
(sjrfc)	<nine, bconst, 0985>	EXT STRING	19S
(sjsend)	<nine, bconst, 0976>	EXT STRING	19T
(sjsubcol)	<nine, bconst, 0975>	EXT STRING	19U
(sjtitle)	<nine, bconst, 0966>	EXT STRING	19V
(sjupdate)	<nine, bconst, 0982>	EXT STRING	19W
(skext)	<nine, bconst, 0878>	EXT STRING	7J10
(skpachk)	<nine, bconst, 0644>	EXT CONSTANT =FALSE	7I2
(spec)	<nine, bconst, 0144>	EXT ADDRESS =1	6A
(sqcls)	<nine, bconst, 0678>	EXT CONSTANT =2	11A2
(sqgnxt)	<nine, bconst, 0679>	EXT CONSTANT =3	11A3
(sqopn)	<nine, bconst, 0221>	EXT CONSTANT =1	11A1
(sqstkn)	<nine, bconst, 0858>	EXT CONSTANT =2	11A4
(sqstks)	<nine, bconst, 028>	EXT ADDRESS =701000B	3A
(srrmax)	<nine, bconst, 0470>	EXT CONSTANT =25	33G2
(sseltyp)	<nine, bconst, 0957>	EXT CONSTANT =0	23B
(sstr)	<nine, bconst, 0722>	EXT STRING	33H22
(stmtv)	<nine, bconst, 0335>	EXT CONSTANT =1	22A
(sub)	<nine, bconst, 0770>	EXT CONSTANT =2	43B
(subdir)	<nine, bconst, 0690>	EXT STRING	28F
(subext)	<nine, bconst, 0880>	EXT STRING	7J8
(suc)	<nine, bconst, 0769>	EXT CONSTANT =3	43C
(sucdir)	<nine, bconst, 0332>	EXT CONSTANT =0	21C
(sucstr)	<nine, bconst, 0686>	EXT STRING	21B
(supcase)	<nine, bconst, 0775>	EXT CONSTANT =4	44C
(svmxlev)	<nine, bconst, 0772>	EXT CONSTANT =64	43F

(swork1)	<nine, bconst, 0157>	EXT CONSTANT =5 7A8
(t2tsbegin)	<nine, bconst, 0826>	EXT 38D
(t2tsend)	<nine, bconst, 0832>	EXT 38F
(tbstr)	<nine, bconst, 0712>	EXT STRING 33H35
(tdtltv)	<nine, bconst, 0349>	EXT CONSTANT =15 22C
(tentil)	<nine, bconst, 0639>	EXT CONSTANT =1 7G3
(textv)	<nine, bconst, 0348>	EXT CONSTANT =14 22N
(tntyp)	<nine, bconst, 0567>	EXT CONSTANT =1 10A2
(tppair)	<nine, bconst, 0417>	EXT CONSTANT =2 32A2
(trnsflag)	<nine, bconst, 0675>	EXT CONSTANT =3 9B3
(tstr)	<nine, bconst, 0727>	EXT STRING 33H23
(ttycoc)	<nine, bconst, 0368>	EXT CONSTANT =10737B3 26A
(tuacc)	<nine, bconst, 0558>	EXT CONSTANT =1000B6 10D4
(txtext)	<nine, bconst, 0196>	EXT STRING 7J6
(txttyp)	<nine, bconst, 0165>	EXT CONSTANT =0 7B1A
(typewriter)	<nine, bconst, 0434>	EXT CONSTANT =0 32F1
(ubrstr)	<nine, bconst, 0714>	EXT STRING 33H31
(ue)	<nine, bconst, 0253>	EXT CONSTANT =15 15I
(up)	<nine, bconst, 0771>	EXT CONSTANT =4 43D
(upcase)	<nine, bconst, 0768>	EXT CONSTANT =40B 43E
(upcompact)	<nine, bconst, 0672>	EXT CONSTANT =3 9A3
(upgbdf)	<nine, bconst, 0800>	EXT ADDRESS =6 17A
(upgtyp)	<nine, bconst, 0215>	EXT CONSTANT =1 8A1
(us)	<nine, bconst, 0252>	EXT CONSTANT =13 15H
(usistr)	<nine, bconst, 0717>	EXT STRING 33H30
(usrexp)	<nine, bconst, 0523>	EXT CONSTANT =100000B 41A3
(usrina)	<nine, bconst, 0525>	EXT CONSTANT =20000B 41A5
(usrlod)	<nine, bconst, 0524>	EXT CONSTANT =40000B 41A4
(usrnot)	<nine, bconst, 0522>	EXT CONSTANT =200000B 41A2
(usroff)	<nine, bconst, 0521>	EXT CONSTANT =400000B 41A1
(ustr)	<nine, bconst, 0728>	EXT STRING 33H24
(utilhost)	<nine, bconst, 0737>	EXT CONSTANT =66 37B
(utilistr)	<nine, bconst, 0739>	EXT STRING 37C
(utilioprtty)	<nine, bconst, 0563>	EXT CONSTANT =1B 10B2
(utilstr)	<nine, bconst, 0860>	EXT STRING 37P
(uustr)	<nine, bconst, 0718>	EXT STRING 33H32
(v)	<nine, bconst, 0322>	EXT ADDRESS =400000B 20A
(vb)	<nine, bconst, 0258>	EXT CONSTANT =25 15N
(vcv)	<nine, bconst, 0325>	EXT ADDRESS =420000B 20D
(vcvend)	<nine, bconst, 0328>	EXT CONSTANT =477750B 20F
(vcvu)	<nine, bconst, 0323>	EXT ADDRESS =400100B 20B
(vcvul)	<nine, bconst, 0324>	EXT ADDRESS =400070B 20C
(ve)	<nine, bconst, 0259>	EXT CONSTANT =27 15O
(vect)	<nine, bconst, 0147>	EXT ADDRESS =4 6D
(vertical)	<nine, bconst, 0695>	EXT CONSTANT =1 33E2
(vexpert)	<nine, bconst, 0226>	EXT ADDRESS =1 12A1
(vfullprompts)	<nine, bconst, 0227>	EXT ADDRESS =3 12A2
(visv)	<nine, bconst, 0345>	EXT CONSTANT =11 22K
(vmax)	<nine, bconst, 0327>	EXT CONSTANT =20000B 20E
(vmultchar)	<nine, bconst, 0229>	EXT ADDRESS =2 12A4
(vstr)	<nine, bconst, 0729>	EXT STRING 33H25
(vtersemode)	<nine, bconst, 0228>	EXT ADDRESS =1 12A3
(vverbsmode)	<nine, bconst, 0230>	EXT ADDRESS =2 12A5
(wacc)	<nine, bconst, 0556>	EXT CONSTANT =40000B6 10D2
(wndw)	<nine, bconst, 0418>	EXT CONSTANT =3 32A5
(wordis)	<nine, bconst, 0365>	EXT CONSTANT =11 25J

(wordtyp)	<nine, bconst, 0357>	EXT CONSTANT =2	25B
(wordv)	<nine, bconst, 0343>	EXT CONSTANT =9	22I
(write)	<nine, bconst, 0211>	EXT CONSTANT =0	7L1
(wstr)	<nine, bconst, 0730>	EXT STRING	33H26
(xcoord)	<nine, bconst, 0593>	EXT CONSTANT =1	32A3
(xopname)	<nine, bconst, 0687>	EXT STRING	28B
(xstr)	<nine, bconst, 0711>	EXT STRING	33H27
(ycoord)	<nine, bconst, 0594>	EXT CONSTANT =2	32A4
(ystr)	<nine, bconst, 0716>	EXT STRING	33H28
(zstr)	<nine, bconst, 0715>	EXT STRING	33H29