(base)	<nine, madata, 015>	EXT	1A3
(cdlnk)	<nine, madata, 06>	EXT	1D
(clpsw)	<nine, madata, 07>	EXT	1E
(ctrlch)	<nine, madata, 09>	EXT	1G
(iprime)	<nine, madata, 012>	EXT CONSTANT =101	1A5
(nlsprname)	<nine, madata, 0109>	EXT STRING	1B
(nlssubs)	<nine, madata, 093>	EXT	1A1
(nwlnk)	<nine, madata, 08>	EXT	1F
(pkgname)	<nine, madata, 066>	EXT STRING	1A2
(pnhasht)	<nine, madata, 014>	EXT	1A6
(psexec)	<nine, madata, 0126>	EXT	1A4
(slink)	<nine, madata, 05>	EXT	1C

```
< NINE, MADATA.NLS;8, >, 13-Apr-78 10:33 LLG ;;;;
FILE madata % <ARCSUBSYS>XL10  to <RELNINE>madata%% <arcsubsys,xl10,>
<arcsubsys,l109,> TO <relnine,madata.rel,>%
   % NLS packages and tools%
      (nissubs) EXTERNAL  %packages automatically loaded%            1A1
         _ (
         $"PROGRAMS" , $programs,
         $"EXEC", $psexec,
         0 ); %end of nlssubs%
      (pkgname) EXTERNAL STRING = "BASE";  %package name for BASE%   1A2
      (base) EXTERNAL %dispatch table (x-routines) for BASE%         1A3
         _ (
         $"ENTRY", $xentry,
         $"EXIT", $xterm,
         $"XAPPEND" , $xappend,
         $"XBREAK" , $xbreak,
         $"XCHKSIMTTY", $xchksimtty,
         $"XCLEAR" , $xclear,
         $"XCLOSE" , $xclose,
         $"XCONNECT", $xconnect,
         $"XCOPY" , $xcopy,
         $"XCREATE" , $xcreate,
         $"XDELETE" , $xdelete,
         $"XENTRY", $xentry,
         $"XEXPAND", $xexpand,
         $"XEXPUNGE", $xexpunge,
         $"XFORCE" , $xforce,
         $"XFREEZE" , $xfreeze,
         $"XFTERM" , $xfterm,
         $"XGETTEXT" , $xgettext,
         $"XGOTO" , $xgoto,
         $"XHANDLE" , $xhandle,
         $"XINIT" , $xinit,
         $"XINSERT" , $xinsert,
         $"XINSSTATEMENT" , $xinsstatement,
         $"XJMPCNT", $xjmpcnt,
         $"XJUMP" , $xjump,
         $"XJUMPADDR" , $xjumpaddr,
         $"XGETJUMPRING" , $xgetjumpring,
         $"XQUIT" , $xfterm,
         $"XJUMPRETURN" , $xjumpreturn,
         $"XLOAD" , $xload,
         $"XLOGOUT", $xlogout,
         $"XMARK" , $xmark,
         % commented out Merge because it never worked
         $"XMERGE" , $xmerge,
         %
         $"XMOUSESPECS" , $xmousespecs,
         $"XMOVE" , $xmove,
         $"XOPEN" , $xopen,
         $"XOUTPUT" , $xoutput,
         $"XCOMPILE" , $xcompile,
         $"XPINSERT", $xpinsert,
         $"XPRINT" , $xprint,
         $"XPROCESS" , $xprocess,
         $"XPRTCC" , $xprtcc,
```

```
        $"XPRTNEXT" , $xprtnext,
        $"XPRTPREV" , $xprtprev,
        $"XPRTSNUM" , $xprtsnum,
        $"XPRTSTMT" , $xprtstmt,
        $"XPUTTEXT" , $xputtext,
        $"XQUITSUBSYS" , $xquitsubsys,
        $"XRELEASE" , $xrelease,
        $"XREPEATSEARCH" , $xrepeatsearch,
        $"XREPLACE" , $xreplace,
        $"XRESET" , $xreset,
        $"XRENUMBER" , $xrenumber,
        $"XSET" , $xset,
        $"XSIMULATE" , $xsimulate,
        $"XSHOW" , $xshow,
        $"XSORT" , $xsort,
        $"XSUBRESOLVE", $xsubresolve,
        $"XSUBSGET", $xsubsget,
        $"XSUBSTITUTE" , $xsubstitue,
        $"XTRANSPOSE" , $xtranspose,
        $"XTRIM", $xtrim,
        $"XUNDELETE", $xundelete,
        $"XUPDATE" , $xupdate,
        $"XVERIFY" , $xverify,
        $"CHECKMORE",  $checkmore,     % checkmore %
        $"UPDCSP",     $updcsp,        % updcsp -- Jump commands %
        $"RSTMORE",    $rstmore,       % rstmor %
        $"HELPINIT",   $helpinit,      % hlpinit %
        $"HELPTERM",   $helpterm,      % hlpinit %
        $"HELPRING",   $helpring,      % xhlpring %
        $"HELPSHOW",   $helpshow,      % helpshow %
        0,0); %marks end of list%
    (psexec) EXTERNAL  %dispatch table (x-routines) in exec
    subsystem%                                                  1A4
        _ ( $"XGOEXEC", $xgoexec, 0, 0);
    (lprime) EXTERNAL CONSTANT = 101;  %size of proc name has table%
                                                                1A5
    (pnhasht) EXTERNAL [lprime]; %procedure name hash table%    1A6
  (nlsprname) EXTERNAL STRING = "NLS";  % process name for DPS trace %
                                                                1B
  (slink) EXTERNAL;  %global referenced by middle-end%         1C
  (cdlnk) EXTERNAL;  %global referenced by middle-end%         1D
  (clpsw) EXTERNAL;  %global referenced by middle-end%         1E
  (nwlnk) EXTERNAL;  %global referenced by middle-end%         1F
  (ctrlch) EXTERNAL;  %global referenced by middle-end%        1G
FINISH
```

(abortecall)	<nine, mconst, 0174>	EXT CONSTANT =10	3C7
(acqpe)	<nine, mconst, 0203>	EXT CONSTANT =13B	3C13A3E
(aloch)	<nine, mconst, 0205>	EXT CONSTANT =11B	3C13A3C
(asis)	<nine, mconst, 099>	EXT CONSTANT =0	3C2
(baddargument)	<nine, mconst, 0171>	EXT CONSTANT =1	3C10
(calpe)	<nine, mconst, 0112>	EXT CONSTANT =7B	3C13A3A
(cbitst)	<nine, mconst, 061>	EXT CONSTANT =5	3A3E
(cblock)	<nine, mconst, 063>	EXT CONSTANT =8	3A3G
(cboole)	<nine, mconst, 058>	EXT CONSTANT =2	3A3B
(cempty)	<nine, mconst, 057>	EXT CONSTANT =1	3A3A
(cindex)	<nine, mconst, 059>	EXT CONSTANT =3	3A3C
(cinteg)	<nine, mconst, 060>	EXT CONSTANT =4	3A3D
(clistt)	<nine, mconst, 064>	EXT CONSTANT =7	3A3H
(clspk)	<nine, mconst, 0207>	EXT CONSTANT =6B	3C13A2B
(crshpg)	<nine, mconst, 087>	EXT CONSTANT =301B	3B16
(crtch)	<nine, mconst, 0116>	EXT CONSTANT =27B	3C13A5A
(crtdt)	<nine, mconst, 0114>	EXT CONSTANT =21B	3C13A4A
(crtev)	<nine, mconst, 0127>	EXT CONSTANT =45B	3C13B5A
(crtlk)	<nine, mconst, 0125>	EXT CONSTANT =41B	3C13B4A
(crtpr)	<nine, mconst, 0121>	EXT CONSTANT =33B	3C13B2A
(crtps)	<nine, mconst, 0108>	EXT CONSTANT =1B	3C13A1A
(crtsp)	<nine, mconst, 0119>	EXT CONSTANT =31B	3C13B1A
(csshpg)	<nine, mconst, 086>	EXT CONSTANT =300B	3B15
(cstrin)	<nine, mconst, 062>	EXT CONSTANT =6	3A3F
(d1sel)	<nine, mconst, 072>	EXT CONSTANT =0	5B
(d2sel)	<nine, mconst, 0215>	EXT CONSTANT =2	5C
(dbox)	<nine, mconst, 076>	EXT CONSTANT =1	3B2
(dcrh)	<nine, mconst, 089>	EXT CONSTANT =3	3B4
(delch)	<nine, mconst, 0192>	EXT CONSTANT =30B	3C13A5B
(deldt)	<nine, mconst, 0197>	EXT CONSTANT =22B	3C13A4B
(delev)	<nine, mconst, 0182>	EXT CONSTANT =46B	3C13B5B
(dellk)	<nine, mconst, 0185>	EXT CONSTANT =42B	3C13B4B
(delpr)	<nine, mconst, 0190>	EXT CONSTANT =34B	3C13B2B
(delps)	<nine, mconst, 0210>	EXT CONSTANT =2B	3C13A1B
(delsp)	<nine, mconst, 0191>	EXT CONSTANT =32B	3C13B1B
(dpsblock)	<nine, mconst, 0149>	EXT	3C16
(dpsrlen)	<nine, mconst, 0152>	EXT	3C18
(dpsrmax)	<nine, mconst, 0153>	EXT	3C19
(drdps)	<nine, mconst, 0168>	EXT CONSTANT =2	3C12C
(ermaio)	<nine, mconst, 048>	EXT CONSTANT =15200B	4B1
(ermbadpcpsim)	<nine, mconst, 053>	EXT CONSTANT =15201B	4B2
(ermblksize)	<nine, mconst, 049>	EXT CONSTANT =15202B	4B3
(ermcnv)	<nine, mconst, 051>	EXT CONSTANT =15203B	4B4
(ermdesc)	<nine, mconst, 046>	EXT CONSTANT =15204B	4B5
(ermentity)	<nine, mconst, 039>	EXT CONSTANT =15205B	4B6
(ermhd)	<nine, mconst, 042>	EXT CONSTANT =15206B	4B7
(ermhov)	<nine, mconst, 041>	EXT CONSTANT =15207B	4B8
(ermhowpcp)	<nine, mconst, 068>	EXT CONSTANT =15210B	4B9
(ermhstor)	<nine, mconst, 043>	EXT CONSTANT =15211B	4B10
(ermillarg)	<nine, mconst, 038>	EXT CONSTANT =15212B	4B11
(erminit)	<nine, mconst, 066>	EXT CONSTANT =15213B	4B12
(ermmsg)	<nine, mconst, 0164>	EXT CONSTANT =15225B	4B22
(ermnargs)	<nine, mconst, 037>	EXT CONSTANT =15214B	4B13
(ermnoproc)	<nine, mconst, 036>	EXT CONSTANT =15215B	4B14
(ermnoroom)	<nine, mconst, 040>	EXT CONSTANT =15216B	4B15
(ermpkgtfull)	<nine, mconst, 065>	EXT CONSTANT =15217B	4B16

```
(ermpraddr)     <nine, mconst, 045>     EXT CONSTANT =15220B 4B17
(ermprname)     <nine, mconst, 044>     EXT CONSTANT =15221B 4B18
(ermptype)      <nine, mconst, 050>     EXT CONSTANT =15222B 4B19
(ermreturn)     <nine, mconst, 067>     EXT CONSTANT =15223B 4B20
(ermstring)     <nine, mconst, 047>     EXT CONSTANT =15224B 4B21
(ermunkown)     <nine, mconst, 073>     EXT CONSTANT =15226B 4B23
(gtdps)         <nine, mconst, 0167>    EXT CONSTANT =3      3C12D
(hippe)         <nine, mconst, 0198>    EXT CONSTANT =20B    3C13A3J
(intpe)         <nine, mconst, 0201>    EXT CONSTANT =15B    3C13A3G
(itdps)         <nine, mconst, 0209>    EXT CONSTANT =3B     3C13A1C
(ivdps)         <nine, mconst, 0104>    EXT CONSTANT =0      3C12A
(lckdt)         <nine, mconst, 0194>    EXT CONSTANT =25B    3C13A4E
(lower)         <nine, mconst, 0178>    EXT CONSTANT =1      3C3
(msgack)        <nine, mconst, 084>     EXT CONSTANT =2      3B13
(msginv)        <nine, mconst, 088>     EXT CONSTANT =1      3B12
(msglen)        <nine, mconst, 075>     EXT CONSTANT =0      3B1
(msgmode)       <nine, mconst, 085>     EXT CONSTANT =3      3B14
(msgp1)         <nine, mconst, 078>     EXT CONSTANT =8      3B6
(msgp2)         <nine, mconst, 079>     EXT CONSTANT =9      3B7
(msgp3)         <nine, mconst, 080>     EXT CONSTANT =10     3B8
(msgp4)         <nine, mconst, 081>     EXT CONSTANT =11     3B9
(msgp5)         <nine, mconst, 082>     EXT CONSTANT =12     3B10
(nopackage)     <nine, mconst, 0176>    EXT CONSTANT =2      3C6
(noprocedure)   <nine, mconst, 0100>    EXT CONSTANT =1      3C5
(ntepe)         <nine, mconst, 0199>    EXT CONSTANT =17B    3C13A3I
(opnpk)         <nine, mconst, 0110>    EXT CONSTANT =5B     3C13A2A
(pagelen)       <nine, mconst, 083>     EXT CONSTANT =512    3B11
(pgdps)         <nine, mconst, 0165>    EXT CONSTANT =5      3C12F
(ptdps)         <nine, mconst, 0166>    EXT CONSTANT =4      3C12E
(rcvch)         <nine, mconst, 0186>    EXT CONSTANT =40B    3C13B3B
(rddt)          <nine, mconst, 0196>    EXT CONSTANT =23B    3C13A4C
(rdypr)         <nine, mconst, 0187>    EXT CONSTANT =36B    3C13B2E
(rdyps)         <nine, mconst, 0129>    EXT CONSTANT =62B    3C13B6A
(relch)         <nine, mconst, 0204>    EXT CONSTANT =12B    3C13A3D
(relpe)         <nine, mconst, 0202>    EXT CONSTANT =14B    3C13A3F
(remlk)         <nine, mconst, 0183>    EXT CONSTANT =44B    3C13B4D
(rrcvchan)      <nine, mconst, 091>     EXT CONSTANT =4      5A
(rrdps)         <nine, mconst, 0169>    EXT CONSTANT =1      3C12B
(rsmpe)         <nine, mconst, 0200>    EXT CONSTANT =16B    3C13A3H
(sbox)          <nine, mconst, 077>     EXT CONSTANT =2      3B3
(scrh)          <nine, mconst, 090>     EXT CONSTANT =4      3B5
(sepps)         <nine, mconst, 0208>    EXT CONSTANT =4B     3C13A1D
(setlk)         <nine, mconst, 0184>    EXT CONSTANT =43B    3C13B4C
(showcnote)     <nine, mconst, 0172>    EXT CONSTANT =2      3C9
(shownote)      <nine, mconst, 0173>    EXT CONSTANT =1      3C8
(siflag)        <nine, mconst, 0155>    EXT                  3C1A
(sigev)         <nine, mconst, 0181>    EXT CONSTANT =47B    3C13B5C
(sipr)          <nine, mconst, 0189>    EXT CONSTANT =35B    3C13B2C
(sndch)         <nine, mconst, 0123>    EXT CONSTANT =37B    3C13B3A
(sopr)          <nine, mconst, 0188>    EXT CONSTANT =53B    3C13B2D
(tstev)         <nine, mconst, 0180>    EXT CONSTANT =50B    3C13B5D
(ubitst)        <nine, mconst, 030>     EXT CONSTANT =8 + lblock 3A2E
(ublock)        <nine, mconst, 031>     EXT CONSTANT =lblock 3A2H
(uboole)        <nine, mconst, 028>     EXT CONSTANT =16 + linteg 3A2(
(uindex)        <nine, mconst, 027>     EXT CONSTANT =8 + linteg 3A2B
(uinteg)        <nine, mconst, 026>     EXT CONSTANT =linteg 3A2A
```

```
(ulist)          <nine, mconst, 033>          EXT CONSTANT =llist  3A2J
(ulkdt)          <nine, mconst, 0193>         EXT CONSTANT =26B    3C13A4F
(unull)          <nine, mconst, 032>          EXT CONSTANT =lnull  3A2I
(upper)          <nine, mconst, 0177>         EXT CONSTANT =2      3C4
(ustrin)         <nine, mconst, 029>          EXT CONSTANT =lstrin 3A2D
(utpblo)         <nine, mconst, 055>          EXT CONSTANT =24 + lblock 3A2(
(utpsbl)         <nine, mconst, 054>          EXT CONSTANT =16 + lblock 3A2F
(vispe)          <nine, mconst, 0206>         EXT CONSTANT =10B    3C13A3B
(vjschan)        <nine, mconst, 0170>         EXT CONSTANT =4      3C11B
(vjspriority)    <nine, mconst, 0148>         EXT                  3C15
(vjubitmap)      <nine, mconst, 0156>         EXT                  3C1B
(vjublock)       <nine, mconst, 0150>         EXT                  3C17
(vjuchan)        <nine, mconst, 0102>         EXT CONSTANT =5      3C11A
(vjusr)          <nine, mconst, 0211>         EXT                  3C14A
(walev)          <nine, mconst, 0179>         EXT CONSTANT =51B    3C13B5E
(wrdt)           <nine, mconst, 0195>         EXT CONSTANT =24B    3C13A4D
```

```
< NINE, MCONST.NLS;2, >, 7-Jul-78 23:11 BLP ;;;;
FILE mconst % <arcsubsys,x110,> TO <relnine,mconst.rel,>%
%This file contains list constants, error codes and constants for the
DPS (MAGIC) version of NLS-9.  The corresponding file for the MSG3, raw
network or shared page versions of NLS-9 is <nine, maconst,>.%
%constants%
   %List type definitions%
      %L10 List - The following branch should be in L10LRP%
         %DECLARE EXTERNAL CONSTANT  descriptor types %
            %
            lnull  = 0 ,   NULL
            linteg = 2 ,   INTEGER
            lstrin = 3 ,   STRING
            llist  = 4 ,   LIST
            lblock = 1 ,   BLOCK
            ldescr = 101 ,   DESCRIPTOR -- never stored as type
            lnewde = 100 ;   NEWDESCRIPTOR -- never stored as type
            %
      %L10 List types including user bits%
         (uinteg) EXTERNAL CONSTANT =  linteg; % 2 INTEGER %          3A2A
         (uindex) EXTERNAL CONSTANT = 8 + linteg; % 10 INDEX%        3A2B
         (uboole) EXTERNAL CONSTANT = 16 + linteg; % 18 BOOLEAN%     3A2C
         (ustrin) EXTERNAL CONSTANT = lstrin; % 3 STRING%            3A2D
         (ubitst) EXTERNAL CONSTANT = 8 + lblock; % 9 BITSTRING%     3A2E
         (utpsbl) EXTERNAL CONSTANT = 16 + lblock; % 17 block containg
         two text pointers and a string%                            3A2F
         (utpblo) EXTERNAL CONSTANT = 24 + lblock; % 25 block containg
         two text pointers%                                         3A2G
         (ublock) EXTERNAL CONSTANT = lblock; % 1 BLOCK%            3A2H
         (unull) EXTERNAL CONSTANT = lnull; % 0 NULL%               3A2I
         (ulist) EXTERNAL CONSTANT = llist; % 4 LIST%               3A2J
      %PCPB36 type codes%
         (cempty) EXTERNAL CONSTANT = 1; %EMPTY%                    3A3A
         (cboole) EXTERNAL CONSTANT = 2; %BOOLEAn %                 3A3B
         (cindex) EXTERNAL CONSTANT = 3; %INDEX%                    3A3C
         (cinteg) EXTERNAL CONSTANT = 4; %INTEGER%                  3A3D
         (cbitst) EXTERNAL CONSTANT =5; %BITSTRING%                 3A3E
         (cstrin) EXTERNAL CONSTANT = 6; %CHARACTER STRING%         3A3F
         (cblock) EXTERNAL CONSTANT = 8; %BLOCK%                    3A3G
         (clistt) EXTERNAL CONSTANT =7; %LIST%                      3A3H
   %msg protocol constants%
      (msglen) EXTERNAL CONSTANT = 0;                               3B1
      (dbox) EXTERNAL CONSTANT = 1;                                 3B2
      (sbox) EXTERNAL CONSTANT = 2;                                 3B3
      (dcrh) EXTERNAL CONSTANT = 3;                                 3B4
      (scrh) EXTERNAL CONSTANT = 4;                                 3B5
      (msgp1) EXTERNAL CONSTANT = 8;                                3B6
      (msgp2) EXTERNAL CONSTANT = 9;                                3B7
      (msgp3) EXTERNAL CONSTANT = 10;                               3B8
      (msgp4) EXTERNAL CONSTANT = 11;                               3B9
      (msgp5) EXTERNAL CONSTANT = 12;                              3B10
      (pagelen) EXTERNAL CONSTANT = 512;                           3B11
      (msginv) EXTERNAL CONSTANT = 1;                              3B12
      (msgack) EXTERNAL CONSTANT = 2;                              3B13
      (msgmode) EXTERNAL CONSTANT = 3;                             3B14
      (csshpg) EXTERNAL CONSTANT = 300B;                           3B15
```

```
   (crshpg) EXTERNAL CONSTANT = 301B;                                    3B16
%dps protocol constants%
   %signon variables%
      (siflag) EXTERNAL  = 204000B;  %sign on flags%                     3C1A
      (vjubitmap) EXTERNAL = %bit map of vjusrs handled.  ABC(bit
      string)%                                                          3C1B
         (000002000002B,  %M,,L%
         cbitst*1B6+15,  %bit string header%
         44001B7);  %prso (1B), pecal (4B), okopk (17B)%
   (asis) EXTERNAL CONSTANT = 0;                                         3C2
   (lower) EXTERNAL CONSTANT = 1;                                        3C3
   (upper) EXTERNAL CONSTANT = 2;                                        3C4
   (noprocedure) EXTERNAL CONSTANT = 1;                                  3C5
   (nopackage) EXTERNAL CONSTANT = 2;                                    3C6
   (abortecall) EXTERNAL CONSTANT = 10;                                  3C7
   (shownote) EXTERNAL CONSTANT = 1;                                     3C8
   (showcnote) EXTERNAL CONSTANT = 2;                                    3C9
   (baddargument) EXTERNAL CONSTANT = 1;                                 3C10
   %psi channels%
      (vjuchan) EXTERNAL CONSTANT = 5;                                   3C11A
      (vjschan) EXTERNAL CONSTANT = 4;                                   3C11B
   %DPS calls%
      (ivdps) EXTERNAL CONSTANT = 0;                                     3C12A
      (rrdps) EXTERNAL CONSTANT = 1;                                     3C12B
      (drdps) EXTERNAL CONSTANT = 2;                                     3C12C
      (gtdps) EXTERNAL CONSTANT = 3;                                     3C12D
      (ptdps) EXTERNAL CONSTANT = 4;                                     3C12E
      (pgdps) EXTERNAL CONSTANT = 5;                                     3C12F
   %VJSYS numbers%
      %remote process manipulation%
         %processes%
            (crtps) EXTERNAL CONSTANT = 1B;                              3C13A1A
            (delps) EXTERNAL CONSTANT = 2B;                              3C13A1B
            (itdps) EXTERNAL CONSTANT = 3B;                              3C13A1C
            (sepps) EXTERNAL CONSTANT = 4B;                              3C13A1D
         %packages%
            (opnpk) EXTERNAL CONSTANT = 5B;                              3C13A2A
            (clspk) EXTERNAL CONSTANT = 6B;                              3C13A2B
         %procedures%
            (calpe) EXTERNAL CONSTANT = 7B;                              3C13A3A
            (vispe) EXTERNAL CONSTANT = 10B;                             3C13A3B
            (aloch) EXTERNAL CONSTANT = 11B;                             3C13A3C
            (relch) EXTERNAL CONSTANT = 12B;                             3C13A3D
            (acqpe) EXTERNAL CONSTANT = 13B;                             3C13A3E
            (relpe) EXTERNAL CONSTANT = 14B;                             3C13A3F
            (intpe) EXTERNAL CONSTANT = 15B;                             3C13A3G
            (rsmpe) EXTERNAL CONSTANT = 16B;                             3C13A3H
            (ntepe) EXTERNAL CONSTANT = 17B;                             3C13A3I
            (hlppe) EXTERNAL CONSTANT = 20B;                             3C13A3J
         %data stores%
            (crtdt) EXTERNAL CONSTANT = 21B;                             3C13A4A
            (deldt) EXTERNAL CONSTANT = 22B;                             3C13A4B
            (rddt) EXTERNAL CONSTANT = 23B;                              3C13A4C
            (wrdt) EXTERNAL CONSTANT = 24B;                              3C13A4D
            (lckdt) EXTERNAL CONSTANT = 25B;                             3C13A4E
            (ulkdt) EXTERNAL CONSTANT = 26B;                             3C13A4F
```

```
        %channels%
           (crtch) EXTERNAL CONSTANT = 27B;                          3C13A5A
           (delch) EXTERNAL CONSTANT = 30B;                          3C13A5B
     %local process manipulation%
        %subprocesses%
           (crtsp) EXTERNAL CONSTANT = 31B;                          3C13B1A
           (delsp) EXTERNAL CONSTANT = 32B;                          3C13B1B
        %processors%
           (crtpr) EXTERNAL CONSTANT = 33B;                          3C13B2A
           (delpr) EXTERNAL CONSTANT = 34B;                          3C13B2B
           (sipr) EXTERNAL CONSTANT = 35B;                           3C13B2C
           (sopr) EXTERNAL CONSTANT = 53B;                           3C13B2D
           (rdypr) EXTERNAL CONSTANT = 36B;                          3C13B2E
        %channels%
           (sndch) EXTERNAL CONSTANT = 37B;                          3C13B3A
           (rcvch) EXTERNAL CONSTANT = 40B;                          3C13B3B
        %locks%
           (crtlk) EXTERNAL CONSTANT = 41B;                          3C13B4A
           (dellk) EXTERNAL CONSTANT = 42B;                          3C13B4B
           (setlk) EXTERNAL CONSTANT = 43B;                          3C13B4C
           (remlk) EXTERNAL CONSTANT = 44B;                          3C13B4D
        %events%
           (crtev) EXTERNAL CONSTANT = 45B;                          3C13B5A
           (delev) EXTERNAL CONSTANT = 46B;                          3C13B5B
           (sigev) EXTERNAL CONSTANT = 47B;                          3C13B5C
           (tstev) EXTERNAL CONSTANT = 50B;                          3C13B5D
           (waiev) EXTERNAL CONSTANT = 51B;                          3C13B5E
        %process%
           (rdyps) EXTERNAL CONSTANT = 62B;                          3C13B6A
  %VJUSRs%
     (vjusr) EXTERNAL =                                               3C14A
        (0,
     %every processor in every subprocess%
        %processes%
           $prso,
        %packages%
           %$inipk, $trmpk,% 0,0,
        %procedures%
           $pecal,
           %$peint, $persm, $peabr, $pente, $pehlp,%  0,0,0,0,0,
        %data stores%
           %$lvrdt, $lrddt, $lwrdt,% 0,0,0,
     % process leader%
        %processes%
           %$okips, $oksps,% 0,0,
        %packages%
           $okopk, %$okcpk,% 0,
        %channels%
           %$okcch, $okdch, $ntlch,% 0,0,0,
           0,0,0,0);


(vjspriority) EXTERNAL = 1; %priority for vjsys psi routine%  3C15
(dpsblock) EXTERNAL = (3B6, 0,0); %for use in doing IVDPS%    3C16
(vjublock) EXTERNAL = (3B6, 0,0); %for use in doing VJUSRS%   3C17
(dpsrlen) EXTERNAL = 5000; %length of block dpsresults %      3C18
```

```
      (dpsrmax) EXTERNAL; %= $dpsresults + dpsrlenend of dpsresults
      block%                                                          3C19
%errors%                                                               4
   % Notification %
   % Program error %
      (ermalo) EXTERNAL CONSTANT = 15200B;                            4B1
      (ermbadpcpsim) EXTERNAL CONSTANT = 15201B;                      4B2
      (ermblksize) EXTERNAL CONSTANT =15202B;                         4B3
      (ermcnv) EXTERNAL CONSTANT = 15203B;                            4B4
      (ermdesc) EXTERNAL CONSTANT = 15204B;                           4B5
      (ermentity) EXTERNAL CONSTANT = 15205B;                         4B6
      (ermhd) EXTERNAL CONSTANT = 15206B;                             4B7
      (ermhov) EXTERNAL CONSTANT = 15207B;                            4B8
      (ermhowpcp) EXTERNAL CONSTANT = 15210B;                         4B9
      (ermhstor) EXTERNAL CONSTANT = 15211B;                         4B10
      (ermillarg) EXTERNAL CONSTANT = 15212B;                        4B11
      (erminit) EXTERNAL CONSTANT = 15213B;                          4B12
      (ermnargs) EXTERNAL CONSTANT = 15214B;                         4B13
      (ermnoproc) EXTERNAL CONSTANT = 15215B;                        4B14
      (ermnoroom) EXTERNAL CONSTANT = 15216B;                        4B15
      (ermpkgtfull) EXTERNAL CONSTANT = 15217B;                      4B16
      (ermpraddr) EXTERNAL CONSTANT = 15220B;                        4B17
      (ermprname) EXTERNAL CONSTANT = 15221B;                        4B18
      (ermptype) EXTERNAL CONSTANT = 15222B;                         4B19
      (ermreturn) EXTERNAL CONSTANT = 15223B;                        4B20
      (ermstring) EXTERNAL CONSTANT =15224B;                         4B21
      (ermmsg) EXTERNAL CONSTANT = 15225B;  %error interfacing with
      MSG3%                                                          4B22
      (ermunkown) EXTERNAL CONSTANT = 15226B;                        4B23
   % Fatal Error %
%misc%
   (rrcvchan) EXTERNAL CONSTANT = 4; %psi channel for MSG runrcv
   primitive%                                                          5A
   (d1sel) EXTERNAL CONSTANT =0;                                       5B
   (d2sel) EXTERNAL CONSTANT =2;                                       5C
FINISH
```

(chntab)	<nine, mdata, 0162>	EXT	2F1A
(dssflag)	<nine, mdata, 0173>	EXT	2H1
(exresi)	<nine, mdata, 0171>	EXT	2A1
(tedpypkg)	<nine, mdata, 0147>	EXT	2D5B
(teident)	<nine, mdata, 0149>	EXT	2D6A
(tepkh)	<nine, mdata, 0142>	EXT	2D2
(teprh)	<nine, mdata, 0144>	EXT	2D4
(tetoolpkg)	<nine, mdata, 0146>	EXT	2D5A
(lev1lc)	<nine, mdata, 0164>	EXT	2F3A
(lev2lc)	<nine, mdata, 0166>	EXT	2F3B
(lev3lc)	<nine, mdata, 0165>	EXT	2F3C
(levtab)	<nine, mdata, 0163>	EXT	2F2A
(wmpkh)	<nine, mdata, 0141>	EXT	2D1
(wmprh)	<nine, mdata, 0143>	EXT	2D3

```
< NINE, MDATA.NLS;3, >, 17-Feb-78 10:07 LLG ;;;;
FILE mdata % <ARCSUBSYS>XL10  to <RELNINE>mdata%% <arcsubsys,xl10,> TO
<relnine,mdata.rel,>%
%writable data%
   %address of shared page message results list%
      (exresl) EXTERNAL;  %address of results from FE%                    2A1
   %data used by gpmiddle%
      DECLARE EXTERNAL stupinfo, toolacnv, toolrcnv, toolcleanup,
      toolinitialize, toolterminate, dpsstatus, smbox, mmbox,
      errorproc, rrcvpage ;
   %package tables%
      DECLARE EXTERNAL
         curpkg , %current package (last opened)%
         prvpkg , %previous package %
         npkgs = 0, %number of packages currently defined%
         pkmaxn = 10, %maximum number of packages%
         pkgs[30 %pkmaxn*pkdesc.SIZE%]; %package descriptors%
      DECLARE EXTERNAL STRING lprocname[20];
   %Package handle for the Frontend.  Used for PCP calls%
      (wmpkh) EXTERNAL ; %  works manager package handle (change when
      we know the correct names) %                                        2D1
      (fepkh) EXTERNAL ; %  front end package handle (change when we
      know the correct names) %                                           2D2
      (wmprh) EXTERNAL ; % works manager process handle %                 2D3
      (feprh) EXTERNAL = 2 ; % front end process handle %                 2D4
      % FE package handles if DPS or callmode it MSG %
         (fetoolpkg) EXTERNAL ;  % tool-package handle or callmode=2%
                                                                          2D5A
         (fedpypkg) EXTERNAL ;  % dpy-package handle or callmode=2 %
                                                                          2D5B
      % FE process handle if DPS or FE mailbox if MSG %
         (feident) EXTERNAL ;                                             2D6A
   %list utility variables%
      DECLARE EXTERNAL
         nlstrecur _ 0; %used to count levels of recurrsion in oplist%
   %pseudo interrupt data%
      %channel table for pseudo interrupts%
         (chntab) EXTERNAL [36];                                          2F1A
      %level table for pseudointerrupts%
         (levtab) EXTERNAL = ($lev1lc, $lev2lc, $lev3lc);                 2F2A
      %storage for pc for different interrupt levels%
         (lev1lc) EXTERNAL ;                                              2F3A
         (lev2lc) EXTERNAL ;                                              2F3B
         (lev3lc) EXTERNAL ;                                              2F3C
   %dummy string used by typeas %
      DECLARE EXTERNAL STRING crdumstr = "
      ";
   %disable show status flag%
      (dssflag) EXTERNAL _ FALSE;  %set to TRUE if no msgs desired%       2H1
   %misc%
      DECLARE EXTERNAL
         minitialized = 0,
         firsttime,
         howpcp = 1,
         mdebug = 0,
         htent =0,
```

```
        htcol =0;
FINISH
```

(pkname)	<nine, middle, 010>	FIELD - 18	3B1A
(pkopen)	<nine, middle, 016>	FIELD - 3	3B1G
(pnlkup)	<nine, middle, 0149>	PROCEDURE	5G
(prime)	<nine, middle, 0221>	PROCEDURE	5J
(psich)	<nine, middle, 029>	FIELD - 9	3B4A
(psichar)	<nine, middle, 0331>	PROCEDURE	8A
(rawa)	<nine, middle, 0705>	EXT CONSTANT =1	3C10A
(rawr)	<nine, middle, 0710>	EXT CONSTANT =4B1	3C10F
(resisize)	<nine, middle, 038>	CONSTANT =30	3C8
(rfi)	<nine, middle, 0707>	EXT CONSTANT =4	3C10C
(rfr)	<nine, middle, 0709>	EXT CONSTANT =2B1	3C10E
(rtnstring)	<nine, middle, 0395>	PROCEDURE	9C
(setchn)	<nine, middle, 0614>	PROCEDURE	8C
(siprr)	<nine, middle, 028>	RECORD	3B4
(traceparams)	<nine, middle, 0566>	PROCEDURE	10A
(unbnded)	<nine, middle, 034>	CONSTANT =-3	3C4

```
< NINE, MIDDLE.NLS;11, >, 21-Mar-78 14:13 LLG ;;;;
FILE middle % using for nls 8.5 <arcsubsys,xl10,> using for nls 9
<arcsubsys,1109,> to <relnine, middle.rel,>%
ALLOW!
%......Declarations......%
   REF newresults;
   % record definitions %
      (pkgdesc) RECORD %Package descriptor%                              3B1
         pkname[18], %addr of L10 string containing package (subsys)
         name%
         pkdir[18], %addr of L10 string containing subsys dir name%
         pkhrea[18], %addr of hash table for this package%
         pkbptr[36], %byte pointer to ASCIZ package name%
         pkhsize[18], %package hash table size%
         pkkeep[3], %flag: true -> don't delete%
         pkopen[3], %flag: true -> this package is open%
         pkhalo[3], %flag: true -> hash table is allocated %
         pkexis[3]; %flag: true -> this entry is in use %
      (pcpb36f) RECORD %PCP 836 FORMAT header record%                    3B2
         pcplen[15], %length or value depending on type%
         pcpd1[3], %unused%
         pcptype[4], %pcp data type%
         pcpd2[13], %unused%
         pcpkey[1]; % pcp key element flag%
      (pagemp) RECORD
         plocked[1],                                                    3B3A
         plink[1];                                                      3B3B
      (siprr) RECORD                                                     3B4
         psich[9], notused[9], page2[9], page1[9];
   % constant declarations %
      (chbmty) CONSTANT = 440700000001B;                                 3C1
      (dps) CONSTANT = 400B; %JSYS to invoke DPS%                        3C2
      (msg) CONSTANT = 215B;  %JSYS to invoke MSG3%                      3C3
      (unbnded) CONSTANT = -3;                                           3C4
      (msg3debug) EXTERNAL  _ TRUE;  %TRUE => debug trace%               3C5
      (disjob) EXTERNAL _ FALSE;  %TRUE => dispatcher created job%       3C6
      (intchannel) EXTERNAL CONSTANT = 5;  %command interrupt channel%
                                                                         3C7
      (reslsize) CONSTANT = 30;  %used to allocated FE results list%     3C8
      % HELP type codes %
         (htypba) EXTERNAL CONSTANT = 1; %bad argument%                  3C9A
         (htypeg) EXTERNAL CONSTANT = 2; %executive grammar help%        3C9B
         (htyptg) EXTERNAL CONSTANT = 3; %tool grammar help%             3C9C
         (htypef) EXTERNAL CONSTANT = 101; %tool grammar help%           3C9D
      %trace bits%
         (rawa) EXTERNAL CONSTANT = 1;  %raw args to local proc%         3C10A
         (atl) EXTERNAL CONSTANT = 2;  %cnvt args'd to local proc%       3C10B
         (rfl) EXTERNAL CONSTANT = 4;  %cnvt results from local proc%
                                                                         3C10C
         (atr) EXTERNAL CONSTANT = 1B1;  %cnvt args to remote proc%
                                                                         3C10D
         (rfr) EXTERNAL CONSTANT = 2B1;  %cnvt results from remote
         proc%                                                           3C10E
         (rawr) EXTERNAL CONSTANT = 4B1;  %raw results from remote
         proc%                                                           3C10F
   % external labels%
```

```
    EXTERNAL intpsi;
% initialization routine %
  (initmiddle) PROCEDURE( procname REF, argconverter REF, cleanup REF,
  resconverter REF, errorprc REF, initialize REF, terminate REF);    4A
    REF exresl, helplist;
    LOCAL i;
    minitialized _ TRUE;
    *lprocname* _ *procname*;
    astruct($lprocname);
    *lprocname*[lprocname.L+1] _ 0;
    npkgs _ 0;
    toolacnv _ &argconverter;
    toolcleanup _ &cleanup;
    toolrcnv _ &resconverter;
    errorproc _ &errorprc ;
    toolinitialize _ &initialize;
    toolterminate _ &terminate;
    lstzone _ $fsblist;
    makezone(lstzone,410,10,$fsblend - $fsblist);
    ipcinit();
    % Initialize global lists %
      &exresl _ getlst(reslsize);  %results from call on FE%
      &helplist _ getlst(reslsize);  %arguments to FE HELP routine%
    RETURN;
    END.
% package support routines %
  (crtlpkg) PROCEDURE( pkgname REF, pkgds REF, dirname REF, keep,
  htable REF, htsize);                                              5A
    LOCAL i=0, pkgh=1,  npairs, psize, dstring REF,  pstring REF,
    hasharea REF, pkg REF;
    &pstring _ getstring(pkgname.L+1,lstzone);
    *pstring* _ *pkgname*;
    astruc(&pstring);
    *pstring*[pstring.L+1] _ 0; %for asciz string%
    &dstring _ getstring(dirname.L,lstzone);
    *dstring* _ *dirname*;
    &pkg _ $pkgs;
    WHILE pkg.pkexis DO
      BEGIN
      IF (pkgh _ pkgh + 1) > pkmaxn THEN ABORT(ermpkgtfull,$"package
      table is full");
      &pkg _ &pkg + pkgdesc.SIZE;
      END;
    IF pkgh > npkgs THEN npkgs _ pkgh;
    pkg.pkname _ &pstring;
    pkg.pkdir _ &dstring;
    pkg.pkbptr _ &pstring+chbmty;
    pkg.pkopen _ pkg.pkexis _ TRUE;
    pkg.pkkeep _ keep;
    WHILE pkgdsfi] DO i _ i+2;
    npairs _ i/2;
    &hasharea _ &htable;
    IF &hasharea AND (htsize >= npairs) THEN
      BEGIN
      psize _ htsize;
      UNTIL prime(psize) DO BUMP DOWN psize;
```

```
      IF (psize <= npairs) OR (psize < 3) THEN &hasharea _
      allohasht(npairs:psize);
      END
   ELSE
      &hasharea _ allohasht(npairs:psize);
   pkg.pkharea _ &hasharea;
   pkg.pkhalo _ NOT ( &htable); %LH non zero implies allocated
   storage for hash table , must free when deleting package%
   pkg.pkhsize _ psize;
   FOR i _ 0 UP UNTIL >= psize DO hashareaCi] _ 0;
   FOR i _ 0 UP 2 UNTIL >= npairs*2 DO
      inspnhash(pkgdsCi], pkgdsCi+1], &hasharea, psize);
   RETURN(pkgh);
   END.
(clopkg) PROCEDURE(pkgh);                                          5B
   LOCAL pkg REF;
   IF pkgh < 1 OR pkgh > pkmaxn THEN RETURN(FALSE);
   &pkg _ $pkgs + ((pkgh-1)*pkgdesc.SIZE);
   IF NOT pkg.pkexis THEN RETURN(FALSE);
   BUMP DOWN pkg.pkopen ;
   RETURN(TRUE);
   END.
(opnpkg) PROCEDURE(pkgh);                                          5C
   LOCAL pkg REF;
   IF pkgh < 1 OR pkgh > pkmaxn THEN RETURN(FALSE);
   &pkg _ $pkgs + ((pkgh-1)*pkgdesc.SIZE);
   IF NOT pkg.pkexis THEN RETURN(FALSE);
   BUMP pkg.pkopen;
   RETURN(TRUE);
   END.
(delpkg) PROCEDURE(pkgh);                                          5D
   LOCAL pkg REF;
   IF pkgh < 1 OR pkgh > pkmaxn THEN RETURN(FALSE);
   &pkg _ $pkgs + ((pkgh-1)*pkgdesc.SIZE);
   IF (NOT pkg.pkexis) OR pkg.pkkeep THEN RETURN(FALSE);
   freestring(pkg.pkname,lstzone);
   freestring(pkg.pkdir,lstzone);
   IF pkg.pkhalo THEN freeblk(pkg.pkharea,lstzone);
   pkg.pkopen _ FALSE;
   pkg.pkhalo _ FALSE;
   pkg.pkexis _ FALSE;
   IF pkgh >= npkgs THEN npkgs _ npkgs - 1;
   RETURN(TRUE);
   END.
(pkgidx) PROCEDURE %Returns package number and dirname of package
whose name identical to the passed string (or 0 if no such package)%
                                                                   5E
   (pname REF);
   LOCAL pkg REF, pkgh=1;
   &pkg _ $pkgs;
   FOR pkgh _ 1 UP UNTIL > npkgs DO
      BEGIN
      IF pkg.pkexis AND (*pname* = *Cpkg.pkname]*) THEN RETURN(pkgh,
      pkg.pkdir);
      &pkg _ &pkg + pkgdesc.SIZE;
      END;
```

```
    RETURN(0);
    END.
(getpkgname)   % CL: ;  get package dirname and filename %
PROCEDURE (pkgh, progname REF);                                    5F
    % Procedure description
       FUNCTION
          Return a string of the format "<dirname>subsname" given a
          package handle
       ARGUMENTS
          pkgh--INT-package handle
          progname--REF-addr of string into which to deposit
          subsystem dir and file names.
       RESULTS
          outcome--BOOL-TRUE if successful.
       NON-STANDARD CONTROL
          none
       GLOBALS
          none
       %
    % Declarations %
       LOCAL pkg REF;
    % calculate package record loc %
       IF pkgh < 1 OR pkgh > pkmaxn THEN
          BEGIN
          *progname* _ NULL;
          RETURN(FALSE);
          END;
       &pkg _ $pkgs + ((pkgh-1)*pkgdesc.SIZE);
    % produce the string %
       *progname* _ "<, *[pkg.pkdir]*, ">, *[pkg.pkname]*;
       astruc(&progname);
    % Return %
       RETURN(TRUE);
    END.

(pnlkup) %returns address of procedure given external name%
PROCEDURE( pkh, pn REF %external name for procedure%);             5G
    % For complete documentation see D.E. Knuths's  "The Art of
    Computer Programming" Volume 3, Sorting and Searching , p. 521%
    %table entry LH = address of procedure, RH = address of string
    containing the name of the procedure%
    LOCAL table REF, pkg REF, i, c, size, inc, j;
       IF pkh < 1 OR pkh > pkmaxn THEN RETURN(FALSE);
       &pkg _ $pkgs + ((pkh-1)*pkgdesc.SIZE);
       IF (NOT pkg.pkexis) OR (pkg.pkopen <= 0) THEN RETURN(FALSE);
       &table _ pkg.pkharea;
       size _ pkg.pkhsize;
       i _ (hash(&pn) MOD size);
       %is this the correct entry?%
       IF table[i] THEN
          %there is an entry, is it the right one?%
          IF *pn* = *[table[i].RH]* THEN RETURN( table[i].LH)
          ELSE
             BEGIN %hash table collision%
             inc _ MIN(MAX(1,pn.L), size-1);
             FOR j _ 1 UP UNTIL >= size DO
```

```
            BEGIN
            IF (i_ i-inc) < 0 THEN i _ i + size;
            IF table[i] = 0 THEN RETURN(0)
               ELSE IF *pn* = *[table[i].RH]* THEN RETURN
               (table[i].LH);
            END;
         ABORT(ermhov, $"hash table overflow",pkh, &pn);
         END
   ELSE RETURN(0);
END.
(inspnhash) %inserts hash table entry for ext callable procedures%
PROCEDURE( pn REF %external name for procedure%, procaddr %address
of procedure%, table REF %hash table for this package%, size %of
hash table%);                                                    5H
   % For complete documentation see D.E. Knuths's  "The Art of
   Computer Programming" Volume 3, Sorting and Searching , p. 521%
   LOCAL i, inc, c, j;
      i _ (hash(&pn) MOD size);
      BUMP htent; %remove after testing hash efficiency%
      %is this the correct entry?%
      IF table[i] THEN
         %there is an entry, is it the right one?%
         IF *pn* = *[table[i].RH]* THEN
            IF table[i].LH = procaddr THEN RETURN( TRUE)
            ELSE ABORT(ermhd, $"attempt to redefine hash table
            entry", &pn)
         ELSE
            BEGIN %hash table collision%
            inc _ MIN(MAX(1,pn.L), size-1);
            BUMP htcol; %remove after testing hash efficiency%
            FOR j _ 1 UP UNTIL >= size DO
               BEGIN
               IF (i_ i-inc) < 0 THEN i _ i + size;
               IF table[i] = 0 THEN
                  BEGIN %Insert entry%
                  table[i].RH _ &pn;
                  table[i].LH _ procaddr;
                  RETURN ( TRUE );
                  END
               ELSE IF *pn* = *[table[i].RH]* THEN
                  IF table[i].LH = procaddr THEN RETURN( TRUE)
                  ELSE ABORT(ermhd, $"attempt to redefine hash table
                  entry", &pn);
               END;
            ABORT(ermhov,$"hash table full");
            END
      ELSE
         BEGIN %Insert entry%
         table[i].RH _ &pn;
         table[i].LH _ procaddr;
         END;
      RETURN (TRUE);
   END.
(allohasht) PROCEDURE( nentries); %allocates a hash table large
enough to hold n entries. The table size is a prime > nentries.
Returns hash table address and actual size%                      5I
```

```
    LOCAL htable REF, size;
    size _ MAX(nentries*4/3, 3);
    UNTIL prime(size) DO BUMP size;
    IF NOT (&htable _ getblk(size,lstzone)) THEN
       ABORT( ermhstor, $"unable to allocate storeage for package
       descriptor hash table",size);
    RETURN(&htable,size);
    END.
(prime) PROCEDURE(n); %Boolean procedure%                              5J
    LOCAL i,q,r;
    IF n < 0 THEN RETURN(FALSE);
    IF n < 4 THEN RETURN(TRUE);
    DIV n/2, q, r;
    IF NOT r THEN RETURN(FALSE);
    FOR i _ 3 UP 2 UNTIL > n DO
       BEGIN
       IF i*i > n THEN RETURN(TRUE);
       DIV n/i, q, r;
       IF NOT r THEN RETURN(FALSE);
       END;
    RETURN(TRUE);
    END.
(initplist) PROCEDURE(pairlst REF LIST, npairs,htable,size);           5K
    LOCAL i, ni, ai, nd, ad;
       FOR i_1 UP UNTIL > npairs DO
          BEGIN
          ni _ (ai _ 2*i) -1;
          nd _ DESCR #pairlst#[ni];
          ad _ ELEM #pairlst#[ai];
          IF (nd.ledtype # linteg AND nd.ledtype # istrin )OR nd.RH =
          0 THEN
             ABORT(ermprname,$"illegal procedure name in package
             descriptor");
          IF NOT ad THEN
             ABORT(ermpraddr,$"illegal procedure address name in
             package descriptor");
          inspnhash( nd.RH ,ad, htable, size);
          END;
       RETURN
    END.
% help call to FE %
   (helpfe)   % LB: ;  issue HELP call to FE %
   PROCEDURE (helpcode, helpmsg REF, arglst REF, reslst REF % =>outcome
   %);                                                                 6A
      % Procedure description
         FUNCTION
            Form argument list and call 'help' procedure in FE via
            'extcall'
            Call argument conversion procedure to convert results
         ARGUMENTS
            helpcode:  help type code - INTEGER
               values:
                  htypba - bad argument
                  htypeg - executive grammar help
                  htyptg - tool grammar help
                  htypef (or > 100) - tool grammar help
```

```
      helpmsg: address of string to be shown to user
      arglst: address of argument list
      reslst:  address for HELP results list
   RESULTS
      outcome:  outcome of FE call
         values:  TRUE if successful; FALSE if failure
   NON-STANDARD CONTROL
      none
   GLOBALS
      none
   %
% Declarations %
   REF helplist;
   LOCAL outcome, resl REF;
% Form "help" procedure argument list %
   IF &helpmsg AND helpmsg.L THEN  %there is message to user%
      #helplist# _
         USE makedesc (uindex, helpcode, FALSE),
         USE makedesc(unull, 0, FALSE),
         *helpmsg*
   ELSE  %no message to user%
      #helplist# _
         USE makedesc (uindex, helpcode, FALSE),
         USE makedesc(unull, 0, FALSE),
         USE makedesc(unull, 0, FALSE);
   IF &arglst AND arglst.L THEN
      #helplist# !_ COPY #arglst#;  %append other arguments%
% Call to FE and convert results into calling procedure's list %
   outcome _ extcall(feident, $"HELP", fetoolpkg, &helplist:
   &resl);
   IF outcome THEN cnvresults(&resl, &reslst);
% Return %
   RETURN(outcome);
END.

(cnvresults)   % LB: ;  convert results list %
PROCEDURE (rawargs REF LIST, conarg REF LIST );                    6B
   % Procedure description
      FUNCTION
         Convert results by calling argument conversion procedure
      ARGUMENTS
         rawargs:  FE result list
         conarg:  Converted result list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL i, coutcome;
      LOCAL STRING cerrstr[100];
   % Loop calling argument conversion %
      FOR i _ 1 UP UNTIL > rawargs.L DO
         BEGIN
```

```
            IF toolacnv THEN
               [toolacnv](&rawargs, &conarg, i, $cerrstr : [coutcome])
            ELSE #conarg#[i] _ COPY #rawargs#[i];
            IF NOT coutcome THEN
               HELP(ermcnv, $cerrstr, i);
            END;
      % Return %
         RETURN;
      END.

% x-routine dispatcher %
   (dispatch) %convert L10 args and call procedure%
   PROCEDURE (pkh, prname REF %string%, rawargs REF LIST, results REF
   LIST );                                                          7A
      LOCAL temp, proc , ltype, ctype , nargs, i, coutcome, poutcome
      ,lpkgh;
      REF proc;
      LOCAL   LIST conarg[15];
      LOCAL STRING cerrstr[100];
         [rubmrk] _ FALSE;  %SHOULD NOT BE RESET HERE BUT WHERE?%
         INVOKE(catdfl);
         % search subsystems (current subsys gets searched first, then
         in order of most recently loaded and then BASE) for procedure.
          each subsystem is a separate package. %
            IF NOT &proc _ pnlkup(curpkg, &prname) THEN
               BEGIN
               lpkgh _ npkgs + 1;
               &proc _ 0;
               UNTIL &proc OR ((lpkgh _ lpkgh - 1 ) <= 0) DO &proc _
               pnlkup(lpkgh, &prname);
               IF lpkgh <= 0 THEN
                  BEGIN
                  *cerrstr* _ "Procedure ", *prname*, " not in dispatch
                  tables";
                  ABORT(ermnoproc,$cerrstr, pkh, &prname);
                  END;
               END;
         nargs _ rawargs.L;
         coutcome _ TRUE;
         FOR i _ 1 UP UNTIL > nargs DO
            BEGIN
            IF toolacnv THEN
               [toolacnv](&rawargs, $conarg, i, $cerrstr : [coutcome])
            ELSE #conarg#[i] _ MOVE #rawargs#[i];
            IF NOT coutcome THEN
               HELP(ermcnv, $cerrstr, i);
            END;
         IF mdebug .A rawa THEN
            BEGIN
            *cerrstr* _ "Raw arguments to local procedure ", *prname*;
            traceparams($cerrstr, &rawargs);
            END;
         IF mdebug .A atl THEN
            BEGIN
            *cerrstr* _ "Converted arguments to local procedure ",
            *prname*;
```

```
            traceparams($cerrstr, $conarg);
            END;
         INVOKE(abrtcmd);
         % Push arguments on stack and call procedure %
            temp _ &results;
            !PUSH S,temp;
            FOR i _ 1 UP UNTIL > nargs DO
               BEGIN
               temp _ ELEM #conarg#[nargs+1-i];
               !PUSH S,temp;
               END;
            temp.LH _ nargs + 1;    temp.RH _ temp.LH;
            proc( : [poutcome]);
            !SUB S,temp;
         IF mdebug .A rf] THEN
            BEGIN
            *cerrstr* _ "Results returned from local procedure ",
            *prname*;
            traceparams($cerrstr, &results);
            END;
         DROP(abrtcmd);
         IF toolcleanup THEN [toolcleanup]();
         #conarg# _ ;
         DROP(catdf1);
         dssflag _ FALSE;  %reset dis. show status flag%
         RETURN [poutcome] ;
         (catdf1)  CATCHPHRASE();                                    7A4R
            CASE SIGNALTYPE OF
               = notetype:
                  CASE SIGNAL OF
                     = return, = unwind :
                        BEGIN
                        DISABLE (catdf1);
                        NULL-LISTS;
                        END;
                  ENDCASE CONTINUE;
            ENDCASE CONTINUE;
         (abrtcmd)  CATCHPHRASE();                                   7A4S
            CASE SIGNALTYPE OF
               = aborttype:
                  BEGIN
                  DISABLE (abrtcmd);
                  IF toolcleanup THEN [toolcleanup] ();
                  dssflag _ FALSE;  %reset dis. show status flag%
                  CONTINUE;
                  END;
            ENDCASE CONTINUE;
      END.
% Pseudo interrupt - arming and processing routines %
   (psichar) % arm a channel for a psi character %
   PROCEDURE ( channel, psiroutine REF, priority, char );              8A
      armpsi( channel, &psiroutine, priority);
      R1.LH _ char;
      R1.RH _ channel;
      !at1();
      RETURN;
```

```
    END.

(armpsi)  %arm pseudo interrupt channel%
PROCEDURE (channel, psiroutine, priority);                              8B
    LOCAL i, mask;
    chntab[channel] _ psiroutine .V (186*priority);
    R1 _ 485;
    R2 _ $chntab;
    R2.LH _ $levtab;
    !sir();
    !eir();
    mask _ 1;
    FOR i _ 0 UP UNTIL >= (35-channel) DO mask _ mask + mask;
    R2 _  mask;
    !aic();
    RETURN;
    END.

(setchn)     % LB:  Set channel for state change or interrupt %
PROCEDURE (jfn, channel);                                               8C
    % Procedure description
       FUNCTION
          Set channel for state change or interrupt.  Channel has
          already been armed.
       ARGUMENTS
          jfn:  jfn for connection
          channel:  channel for connection state interrupt
       RESULTS
          proc-value
       NON-STANDARD CONTROL
          none
       GLOBALS
          none
       %
    % Declarations %
    % Set channel for connection state change or interrupt%
       R1 _ jfn;
       R2 _ 24B;
       R3 _ channel;
       !mtopr();
    % Return %
       RETURN;
    END.

(introutine) %DO NOT CALL THIS ROUTINE -- PSI ROUTINE%
PROCEDURE;                                                              8D
    (intpsi):  % command interrupt psi routine %                        8D1
       % here for ^O. interrupt when there And INS/INR on receive
       connection (raw network) or interrupt on command interrupt
       channel (shared page) or really ^O (one-fork).  set flag
       inderectly through 'rubmrk'. flag is checked by BE routines. %
       %save ac's%
          psiac1 _ R1;
          R1 _ $psiacs;
          !BLT R1, psiace;
       % set ^O global %
```

```
            [rubmrk] _ TRUE;
         % restore ac's %
            R1.LH _ $psiacs;
            R1.RH _ 0;
            !BLT R1, 17B;
            R1 _ psiac1;
         % return to interrupted code %
            !debrk();
      END.
% list support routines %
   (mktpblk) PROCEDURE(litstr REF %literal string%);
   %allocates a block and fills it with a pair of textpointers to the
   beginning and the end of a copy of the literal string . The
   textpointers point to the beginning and end of the copy of the
   literal string%                                                    9A
      LOCAL strlen, blklen;
      LOCAL lsptr REF, blkptr REF;
      LOCAL TEXT POINTER tpstart, tpend;
         blklen _  (litstr.L+4)/5 + 5;
         &blkptr _ getblk( blklen, lstzone);
         &lsptr _ &blkptr + 4;
         lsptr.M _ litstr.L;
         *lsptr* _ NULL;
         *lsptr* _ *litstr*;
         blkptr _ &lsptr;
         blkptr.stastr _ 1;
         blkptr[1] _ 1;
         blkptr[2] _ blkptr;
         blkptr[3] _ litstr.L+1;
         RETURN(makedesc(utpsbl,&blkptr) );
      END.
   (makedesc) PROCEDURE (type, value, aloflg);
   %returns an L10 List decsciptor%                                   9B
      LOCAL des;
      des _ 0;
      des.ledval _ value;
      des.ledubv _ type;
      IF aloflg THEN des.ledalo _ TRUE;
      RETURN(des);
      END.
   (rtnstring) PROCEDURE (strp REF %string%);
   %allocates a sting of length strp.L and copies strp. Returns a List
   Element Descriptor for the new string%                             9C
      LOCAL led REF, wsize;
         wsize _ (strp.L + 4) / 5;
         &led _ alloblk(wsize+1);
         led.L _ led.M _ strp.L;
         blkxfr(&strp + 1, &led + 1,wsize);
         &led _ makedesc(ustrin,&led,TRUE);
         RETURN(&led);
      END.
   (getpstring) PROCEDURE (selist REF, astring REF);                  9D
      % given an address of a list whse second element is a pointer to
      two text pointers to a string, and the address of a string, puts
      the pointed to string into the addressed string. %
      LOCAL ptr1 REF, ptr2 REF;
```

```
   &ptr1 _ ELEM #selist#[tppair];
   &ptr2 _ &ptr1+d2sel;
   *astring* _ ptr1 ptr2;
   RETURN;
   END.

(alobik) PROCEDURE (size);
%allocates a block of at least size words in the list allocation
zone, lstzone. Returns the address of the first usable word of the
block, generates a signal if the allocation fails%                    9E
   LOCAL blkaddr, ok;
      blkaddr _ getblk(size,lstzone : [ok] );
      IF NOT ok THEN ABORT(ermalo, $"list storage allocation full",
      size);
      RETURN(blkaddr);
   END.
(oplist) PROCEDURE(ladr REF LIST);
% prints the contents of a list to the terminal%                      9F
   LOCAL i;
   LOCAL STRING as[300];
      *as* _ NULL;
      FOR i _ 1 UP UNTIL > MIN(2*nlstrecur,30) DO
         *as* _ *as* , '  ';
      *as* _ *as* , " List at ", STRING(&ladr,8), ' ,STRING(ladr.L),
      '/ , STRING(ladr.M),  " :" ;
      mtypeas($as);
      BUMP nlstrecur;
      FOR i _ 1 UP UNTIL >ladr.L DO oplelem(&ladr,i);
      nlstrecur _ nlstrecur -1;
      RETURN;
   END.
(oplelem) PROCEDURE(ladr REF LIST , indx);
% prints the contents of a list elem to the terminal%                9G
   LOCAL type, utype, i, ltp1 REF, ltp2 REF;
   LOCAL STRING as[500];
      *as* _ NULL;
      FOR i _ 1 UP UNTIL > MIN(2*nlstrecur,30) DO
         *as* _ *as* , '  ';
      *as* _ *as* , STRING(indx) , ');
      utype _ lreadb(&ladr,indx );
      type _ ( DESCR #ladr#[indx]).ledtyp ;
      CASE type OF
         = inull: *as* _  *as*, "NULL";
         = linteg: *as* _  *as*, "INTEGER";
         = lstrin: *as* _  *as*, "STRING";
         = lblock: *as* _  *as*, "BLOCK";
         = llist: NULL;
      ENDCASE *as* _  *as*, "illegal list element type";
      CASE utype OF
         = unull: NULL;
         = uinteg: *as* _ *as* ," " , STRING(ELEM #ladr#[indx]), '[
         , STRING(ELEM #ladr#[indx],8), '] ;
         = uindex: *as* _ *as* , "(INDEX) ", STRING(ELEM
         #ladr#[indx]), '[, STRING(ELEM#ladr#[indx],8), '];
         = uboole:
            BEGIN
```

```
        *as* _ *as* , "(BOOLEAN) ";
        IF ELEM #ladr#[indx] THEN *as* _ *as* , "TRUE"
           ELSE *as* _ *as* , "FALSE";
        END;
     = ustrin:
        *as* _ *as*, " at ", STRING(ELEM #ladr#[indx],8),
           " is (", STRING([ELEM #ladr#[indx]].L), "/,
           STRING([ELEM #ladr#[indx]].M), ") =",
           *[ELEM #ladr#[indx]]*;
     = ublock:
        BEGIN
        *as* _ *as*, " at ", STRING(ELEM #ladr#[indx],8),
           " length (", STRING([ELEM #ladr#[indx]-1].blklength),
           ") =";
        FOR i _ 0 UP UNTIL >= [ELEM #ladr#[indx]-1].blklength DO
           *as* _ *as* , " ", STRING([ELEM
           #ladr#[indx]+i].LH,8), ",," ,
              STRING([ELEM #ladr#[indx]+i].RH,8);
        END;
     = utpsbl:
        BEGIN
        *as* _ *as*, "(LITERAL) at ", STRING(ELEM
        #ladr#[indx],8),
           " length (", STRING([ELEM #ladr#[indx]-1].blklength),
           ") =";
        &ltp1 _ ELEM #ladr#[indx];
        &ltp2 _ &ltp1 + d2sel;
        IF ltp1 = ltp2 THEN
           *as* _ *as*, " [", ltp1 ltp2, "];
        END;
     = utpblo:
        BEGIN
        *as* _ *as*, "(ENTITY) at ", STRING(ELEM
        #ladr#[indx],8),
           " length (", STRING([ELEM #ladr#[indx]-1].blklength),
           ") =";
        FOR i _ 0 UP UNTIL >= 2*d2sel DO
           *as* _ *as*,  " ", STRING([ELEM
           #ladr#[indx]+i].LH,8), "," ,
              STRING([ELEM #ladr#[indx]+i].RH,8);
        &ltp1 _ ELEM #ladr#[indx];
        &ltp2 _ &ltp1 + d2sel;
        IF ltp1 = ltp2 THEN
           *as* _ *as*, " [", ltp1 ltp2, "];
        END;
     = ubitst:
        BEGIN
        *as* _ *as*, "(BITSTR) at ", STRING(ELEM
        #ladr#[indx],8),
           " length (", STRING([ELEM #ladr#[indx]-1].blklength),
           ") =",
           " Bitcount= ", STRING([ELEM #ladr#[indx]]);
        FOR i _ 1 UP UNTIL >= [ELEM #ladr#[indx]-1].blklength DO
           *as* _ *as* , " ", STRING([ELEM
           #ladr#[indx]+i].LH,8), ",," ,
              STRING([ELEM #ladr#[indx]+i].RH,8);
```

```
            END;
         = ulist: NULL;
      ENDCASE *as* _ *as*,  "illegal element type user bits";
      mtypeas($as);
      IF type = llist THEN oplist( ELEM #ladr#[indx]);
      RETURN;
   END.
(copyasciz) %copy ASCIZ string to L10 string%
PROCEDURE (how, length, bp, string REF);                                9H
   LOCAL u, l, char;
   CASE how OF
      = lower:
         BEGIN
         how _ 40B;
         l _ 'A;
         u _ 'Z;
         END;
      = upper:
         BEGIN
         how _ -40B;
         l _ 'a;
         u _ 'z;
         END;
      ENDCASE %asis% how _ l _ u _ 0;
   WHILE (length := length - 1) DO
      BEGIN
      CASE (char _ ^bp) OF
         = 0: IF length < 0 THEN RETURN;
            %length was unknown -- string to terminate with null
            character%
         IN [l, u] : char _ char + how;
         ENDCASE;
      *string* _ *string*, char;
      END;
   RETURN;
   END.

(l10toasciz)  %converts l10 string to ASCIZ%
PROCEDURE (src REF, dst REF %=>stringlength in words%);                 9I
   LOCAL dumsrc REF, strlen;
   &dumsrc _ IF src.L < src.M THEN &src
      ELSE getstring(src.L + 1, lstzone);
   *dumsrc* _ *src*,0;
   blkxfr(&dumsrc + 1, &dst, (strlen_ (dumsrc.L + 4)/5));
   IF dumsrc.M # src.M THEN freestring(&dumsrc, lstzone);
   RETURN(strlen);
   END.
(asczto1s) PROCEDURE (string ,bp);                                      9J
   %convert ASCIZ byte pointer to NLS string%
   %----------%
   REF string;
   LOCAL char;
   string.L _ 0;
   UNTIL (char _ ^bp) = 0 DO *string* _ *string*, char;
   RETURN
   END.
```

```
(ascizsize) PROCEDURE (bp REF % => length%);                          9K
   %returns the length of asciz string in chars%
   LOCAL CONSTANT strmax = 7777778;
   R2 _ 0;
   R3 _ &bp;
   DO
      BEGIN
      !ILDB   R1, (R3);
      IF R1 THEN BUMP R2;
      END
   UNTIL NOT R1 OR R2.RH > strmax;
   RETURN(R2);
   END.
% miscellaneous support routines %
(traceparams) PROCEDURE %traces parameters passed to/from external
procedures%                                                           10A
   (astrng,  %addr of a string to display prior to trace%
   ladr);  %addr of list holding paramenters%
   mtypeas($" ");
   mtypeas(astrng);
   oplist(ladr);
   mtypeas($" ");
   RETURN;
   END.
(idle) PROCEDURE; %wait for an interrupt%                             10B
   !wait();
   RETURN;
   END.
%(err) PROCEDURE (errno);
   ABORT(ermunknown,errno)
   END.%
(mtypeas) PROCEDURE (astrng); %Type a-string on tty%                  10D
   %-----------------%
   REF astrng;
   IF NOT astrng.L THEN RETURN;
   !sout(101B, chbmty + &astrng, -astrng.L);
   !sout(101B, chbmty + $crdumstr, -1);
   RETURN;
   END.

(blkxtr) %copy block%
PROCEDURE (src REF, dst REF, length);                                 10E
   %verify size of block%
      IF length < 0 THEN ABORT (ermblksize, $"negative block size
      detected in blkxfr");
   %copy block%
      IF length AND &dst THEN
         BEGIN
         R1 _ &dst; R1.LH _ &src;
         R2 _ &dst + length - 1;
         !BLT R1,(R2);
         END;
   %return%
      RETURN;
      END.
```

```
   (btypestr) PROCEDURE (astrng); %Type a-string on tty%                10F
      %-----------------%
      REF astrng;
      IF NOT astrng.L THEN RETURN;
      !sout(101B, chbmty + &astrng, -astrng.L);
      RETURN;
      END.

   (coreturn) PROCEDURE ( arglst REF, rtnlst REF LIST);                 10G
      IF NOT howpcp THEN RETURN;
      RETURN;
      END.
FINISH
```

(callct)	<nine, msg3data, 090>	EXT	5C
(connok)	<nine, msg3data, 0138>	EXT	5D9
(debug)	<nine, msg3data, 0170>	EXT	5D1
(erraddr)	<nine, msg3data, 0153>	EXT	5F2
(errjfn)	<nine, msg3data, 0152>	EXT	5F1
(errstate)	<nine, msg3data, 0144>	EXT	5D10
(fhandle)	<nine, msg3data, 0160>	EXT	5E2
(fhost)	<nine, msg3data, 0105>	EXT	5D4
(frsckt)	<nine, msg3data, 0107>	EXT	5D6
(fsskt)	<nine, msg3data, 0106>	EXT	5D5
(inrcvbuf)	<nine, msg3data, 012>	EXT	5A1
(inrcvname)	<nine, msg3data, 059>	EXT	5B
(inrcvpb)	<nine, msg3data, 015>	EXT	5A4
(irbsiz)	<nine, msg3data, 09>	EXT CONSTANT =1	4B
(msginputjfn)	<nine, msg3data, 098>	EXT	5D2
(msgoutputjfn)	<nine, msg3data, 099>	EXT	5D3
(pagelen)	<nine, msg3data, 07>	EXT CONSTANT =512	4A
(rcvchecking)	<nine, msg3data, 0161>	EXT	5E3
(rcvrdy)	<nine, msg3data, 0108>	EXT	5D7
(ropnd)	<nine, msg3data, 0145>	EXT	5D11
(rrfcs)	<nine, msg3data, 0146>	EXT	5D12
(sbsiz)	<nine, msg3data, 010>	EXT CONSTANT =1	4C
(sendpb)	<nine, msg3data, 014>	EXT	5A3
(sndbuf)	<nine, msg3data, 013>	EXT	5A2
(sndchannel)	<nine, msg3data, 0159>	EXT	5E1
(sndrdy)	<nine, msg3data, 0135>	EXT	5D8
(sopnd)	<nine, msg3data, 0147>	EXT	5D13
(srfcs)	<nine, msg3data, 0148>	EXT	5D14

```
< NINE, MSG3DATA.NLS;3, >, 13-Apr-78 14:02 LLG ;;;;
FILE msg3data % (arcsubsys,x110,) ( relnine, msg3data.rel,) %
%MUST LOAD THIS FILE AT PAGE 300 (3000000)%
%DO NOT PUT ANYTHING BEFORE THE SEND AND RECEIVE BUFFERS%
%buffer sizes%
   (pagelen)  EXTERNAL CONSTANT = 512;                                    4A
   (irbsiz)  EXTERNAL CONSTANT = 1;  %receive bufter size should be 2 *
    pagelen  for FE on PDP 11%                                            4B
   (sbsiz)  EXTERNAL CONSTANT = 1;  %send buffer size should be 2 *
   pagelen  for FE on PDP 11%                                             4C
%writable data%
   %buffers inrcvbuf and sndbuf MUST BE ON PAGE BOUNDARY%
      (inrcvbuf) EXTERNAL [irbsiz]; %receive buffer for inline calls%
                                                                          5A1
      (sndbuf) EXTERNAL [sbsiz]; %send buffer for most calls%             5A2
      (sendpb) EXTERNAL [10]; %parameter block area for most sends%       5A3
      (inrcvpb) EXTERNAL [10]; %param block%                              5A4
   (inrcvname) EXTERNAL [10]; %process name areas for inline calls%        5B
   (callct) EXTERNAL = 0;  %number of outstanding calls%                   5C
   %direct connection data%
      (debug) EXTERNAL _ 0;  %switch for debugging output%                 5D1
      (msginputjfn) EXTERNAL;  %input jfn - data from FE%                  5D2
      (msgoutputjfn) EXTERNAL;  %output jfn - data to FE%                  5D3
      (fhost) EXTERNAL;  %foreign host number for FE%                      5D4
      (fssckt) EXTERNAL;  %foreign socket number for FE send %             5D5
      (frsckt) EXTERNAL;  %foreign socket number for FE receive %          5D6
      (rcvrdy) EXTERNAL _ FALSE;  %receive connection openf flag%          5D7
         %TRUE when OPENF is done - open may not have completed%
      (sndrdy) EXTERNAL _ FALSE;  %send connection openf flag%             5D8
         %TRUE when OPENF is done - open may not have completed%
      (connok) EXTERNAL _ FALSE;  %TRUE if both connections opened%        5D9
         %Reset to FALSE if anything goes wrong on either connection%
      (errstate) EXTERNAL _ FALSE;  %TRUE if anything goes wrong on
      either connection%                                                  5D10
      (ropnd) EXTERNAL _ FALSE;  %TRUE if receive connection opened%
                                                                          5D11
      (rrfcs) EXTERNAL _ FALSE;  %TRUE if receive connection in wait%
                                                                          5D12
      (sopnd) EXTERNAL _ FALSE;  %TRUE if send connection opened%         5D13
      (srfcs) EXTERNAL _ FALSE;  %TRUE if send connection in wait
      state%                                                              5D14
   %shared page data%
      (sndchannel) EXTERNAL;  %channel for messages to FE%                5E1
      (fhandle) EXTERNAL;  %FE fork handle%                               5E2
      (rcvchecking) EXTERNAL = FALSE;  %TRUE if at wait for psi from
      FE%                                                                 5E3
      %rcvrdy - defined above.  TRUE if unprocessed message from FE%
   %error log data%
      (errjfn) EXTERNAL = 0;  %jfn for error log file%                    5F1
      (erraddr) EXTERNAL = 0;  %address of error message%                 5F2
FINISH
```

(dpsrend)	<nine, mshared, 07>	EXT	2C
(dpsresults)	<nine, mshared, 05>	EXT	2B

```
< NINE, MSHARED.NLS;3, >, 6-AUG-76 09:20 EKM ;;;;
FILE mshared % <arcsubsys,x110,> TO <relnine,mshared.rel,>%
%dps global data%
   %do not interchange the following two declarations!%
   (dpsresults) EXTERNAL [10000B]; %huge block for DPS to use for
   results%                                                          2B
   (dpsrend) EXTERNAL ;                                              2C
FINISH
```

```
< NINE, NLSACNV.NLS;10, >, 10-May-78 09:30 SKO ;;;;
FILE nlsacnv % <ARCSUBSYS>XL10 to <RELNINE>nlsacnv %% (arcsubsys,x110,)
(RELNINE,nlsacnv.rel,) %
   (cnvarg) PROCEDURE (raw REF LIST , carg REF LIST, indx, errmsg REF
   %string%);                                                          1A
      LOCAL temp, type, argtype, tpblk REF , lstptr REF, slist REF,
      delimiter REF, bugs, da REF, delimstr, litstr, ent, freeit _
      TRUE;
      LOCAL TEXT POINTER rtptr;
         IF indx = 1 THEN  %initialize window selection occurance flag%
            wsflag _ FALSE;
         CASE (type _ lreadb( &raw,indx) ) OF
            =unull, = uinteg, = uboole, = ustrin, = ubitst:
               #carg#[indx] _ COPY #raw#[indx];
            = ulist:
               BEGIN
               &lstptr _ ELEM #raw#[indx];
               CASE lreadb(&lstptr,1) OF
                  = uinteg: #carg#[indx] _ COPY #raw#[indx];
                  = ulist : #carg#[indx] _ COPY #raw#[indx];
                  = uindex :
                     BEGIN
                     argtype _ ELEM #lstptr#[1];
                     IF (lstptr.L = 2) AND (lreadb(&lstptr,2) = ustrin)
                     THEN
                        BEGIN  %This is either a LSELECTOR or a Command
                        word%
                        %add link delimiters if necessary%
                           litstr _  ELEM #lstptr#[2];
                           CASE argtype OF
                              =30: %link%
                                 delimstr _ littolnk(FALSE,litstr);
                              = 7, %oldfilename% = 6: %newfilename%
                                 delimstr _ littolnk(TRUE,litstr);
                              = 121, = 13: %ident, identlist%
                                 delimstr _ identinterp(litstr);
                              ENDCASE
                                 BEGIN
                                 freeit _ FALSE; % don't free delimstr
                                 %
                                 delimstr _ litstr;
                                 END;
                           #carg#[indx] _ LIST (
                              COPY #lstptr#[1] ,
                              USE mktpblk(delimstr));
                           IF freeit THEN freestring(delimstr, $dspblk);
                           END
                     ELSE
                        CASE argtype   OF
                           = 11:  %Window selection%
                              BEGIN
                              %check window id%
                                 findwa (wsvalu _ ELEM #lstptr#[2]);
                              wsflag _ TRUE;  %occurance flag%
                              cwndow _ wsvalu;
                              #carg#[indx] _ wsvalu;
```

```
                END;
            = 1: % Address , viewspecs or leveladjust%
                BEGIN
                ent _ ELEM #lstptr#[2]; %entity type%
                &slist _ ELEM #lstptr#[3];
                CASE ent OF
                   = 48: %viewspecs%
                      BEGIN
                      %Check window id%
                         findwa(temp _ ELEM #slist#[3]);
                      cwndow _ temp ;
                      #carg#[indx] _ USE cnvvsp(ELEM
                      #slist#[2], cwndow);
                      END;
                   = 49: %levaladjust%
                      #carg#[indx] _ cnvladj(ELEM
                      #slist#[2]);
                   ENDCASE  %address%
                      BEGIN
                      bugs _ lstptr.L - 2;
                      IF bugs > 2 THEN
                         ABORT(ermillarg, $"More than 2
                         selections in adress
                         expression");
                      &delimiter _ cnvdlm(ent);
                      &tpblk _ getblk (4,lstzone);
                      %Process first address%
                         CASE ELEM #slist#[1] OF
                            = 1 : % ASELECTOR%
                               BEGIN
                               cwndow _ ELEM #slist#[3];
                               &da _ dsparea(cwndow);
                               rtptr _ da.dacsp;
                               rtptr[1] _ da.daccnt;
                               cnvaddr(&tpblk, $rtptr,
                               ELEM #slist#[2], &da);
                               END;
                            = 2 : % PSELECTOR%
                               cnvbug( &slist, &tpblk);
                            ENDCASE;
                      %Process second address if there is
                      one%
                         IF bugs > 1 THEN
                            BEGIN  %get second address%
                            &slist _ ELEM #lstptr#[4];
                            CASE ELEM #slist#[1] OF
                               = 1 : % ASELECTOR%
                               BEGIN
                               &da _ dsparea(cwndow);
                               cnvaddr(&tpblk + d2sel,
                               &tpblk, ELEM #slist#[2],
                               &da);
                               END;
                               = 2 : % PSELECTOR%
                               cnvbug( &slist, &tpblk +
                               d2sel);
```

```
                                        ENDCASE;
                                    delimiter(&tpblk,&tpblk+d2sel
                                    ,&tpblk,&tpblk+d2sel);
                                    END
                                ELSE
                                    BEGIN
                                    tpblk[2] _ tpblk;
                                    tpblk[3] _ tpblk[1];
                                    delimiter(&tpblk, &tpblk,
                                    &tpblk+d2sel)
                                    END;
                                #carg#[indx] _ LIST(
                                    ent,
                                    USE makedesc(utpblo,&tpblk
                                    ,TRUE), cwndow);
                                END;
                            END;
                        ENDCASE #carg#[indx] _ COPY #raw#[indx];
                    END;
                ENDCASE #carg#[indx] _ COPY #raw#[indx];
            END;
        ENDCASE ABORT (ermillarg, $"Illegal argument type");
    %If last argument in this command, set override value for
    current window-id if a window-id entity type occurred as one
    of the arguments for this command%
        IF indx = raw.L AND wsflag THEN
            cwndow _ wsvalu;
    RETURN (TRUE) ;
END.
(cnvdlm) PROCEDURE  (ent  %entity type%);                                    1B
    %Determine delimiter routine for ASELECTOR or PSELECTOR%
    LOCAL addr;
    %what happens to this stuf in NSW ?%
    slink _ cdlnk _ clpsw _ nwlnk _ FALSE;
    CASE ent OF
        % STRUCTURAL ENTITIES %
            = 26 %- branch -%:
                addr _ $stdr;
            = 27 %- group -%:
                addr _ $grpdr;
            = 28 %- plex -%:
                addr _ $plxdr;
            = 29, = 5 %- statement -%:
                addr _ $stdr;
        % TEXTUAL ENTITIES %
            = 2 %- character -%:
                addr _ $cdr;
            = 34 %- controlchar -%:
                BEGIN
                addr _ $tdr;
                ctrlchar _ TRUE; % special echos %
                END;
            = 11 %- invisible -%:
                addr _ $idr;
            = 30 %- link -%:
                BEGIN
```

```
            slink _ TRUE;
            cdlnk _ TRUE;
            addr _ $ldr;
            END;
         = 36 %- directory -%:
            BEGIN
            slink _ TRUE;
            addr _ $vdr;
            END;
         = 10 %- password -%:
            BEGIN
            clpsw _ TRUE;
            addr _ $vdr;
            END;
         = 8 %- number -%:
            addr _ $ndr;
         = 9 %- real number -%:
            % there should be a new delimiter written for real
            numbers%
            addr _ $ndr;
         = 1 %- text -%:
            addr _ $tdr;
         = 4 %- visible -%:
            addr _ $vdr;
         = 3 %- word -%:
            addr _ $wdr;
      % MISC. ENTITIES %
         = 37 %- file -%:
            BEGIN
            slink _ cdlnk _ TRUE;
            addr _ $flndr;
            END;
         = 6 %- newfilelink -%:
            BEGIN
            slink _ cdlnk _ nwlnk _ TRUE;
            addr _ $ldr;
            END;
         = 7 %- oldfilelink -%:
            BEGIN
            slink _ cdlnk _ TRUE;
            addr _ $ldr;
            END;
         = 32 %- name -%:
            addr _ $nmdr;
         = 38 %- ident -%:
            addr _ $iddr;
         = 39 %- identlist -%:
            addr _ $idldr;
         = 33 %- edge -%:
            addr _ 0;
         = 40 %- marker -%:
            addr _ $wdr;
      ENDCASE ABORT(ermentity, $"Unknown entity selection type -
      cnvdlm");
   RETURN ( addr);
   END.
```

```
(cnvaddr) PROCEDURE  %Convert address selection to text pointer%   1C
   (dstptr REF %address into which to store resulting text pointer%,
   rtptr REF  %address of text pointer relative to which address
   will be evaluated%,
   adstr REF %address string%,
   da REF  %address of current display area%);
   LOCAL TEXT POINTER  z1, z2  %working text pointers%;
   LOCAL delimstr REF;
   dstptr _ rtptr;
   dstptr[1] _ rtptr[1];
   slink _ cdlnk _ TRUE;
   &delimstr _ littolnk(FALSE, &adstr);
   FIND SF(*delimstr*) ^z1 SE(*delimstr*) ^z2;
   IF da.daempty AND NOT FIND SF(z1) $(SP/TAB) ( "( / "< / "--" )
   THEN err($"No file currently loaded.");
   cspvs _ caddexp($z1, $z2, &da, &dstptr : cspvs[1], cspcacode,
   cspusqcod);
   RETURN;
   END.
(cnvbug) PROCEDURE (bpoint REF LIST %point%, tptr REF %text
pointer%);                                                          1D
   %Converts a bug to a text pointer.  Knows the format of the
   display area records.%
   LOCAL block, da REF, entry REF, rflag, festid, felsid, charct;
   cwndow _ ELEM #bpoint#[2];  %Set current window-id global%
   &da _ lda();
   %Find matching front-end stid%
      festid _ ELEM #bpoint#[3];
      IF NOT (block _ da.dastrt  %Head of string table%) THEN
         ABORT(ermillarg, $"Address not found in display area -
         cnvbug");
      &entry _ block + dspbhl;  %first entry%
      WHILE entry.strngid # festid DO
         IF NOT dgetnxt(&entry, block, forward: &entry, block) THEN
            ABORT(ermillarg, $"Address not found in display area -
            cnvbug");
   %Find matching front-end lsid%
      felsid _ ELEM #bpoint#[4];
      WHILE entry.stlsid # felsid DO
         IF NOT dnxtls(&entry, block: &entry, block) THEN
            ABORT(ermillarg, $"Address not found in display area -
            cnvbug");
   %Form text pointer%
      tptr _ entry.ststid;
      IF entry.stcbug THEN
         BEGIN  %bug each char%
         %Check character position%
            IF (charct _ ELEM #bpoint#[5]) > slngth(entry.stbps,
            entry.stbpe) THEN
               ABORT(ermillarg, $"Address not found in display area
               - cnvbug");
         tptr[1] _ entry.stccnt + charct - 1
         END
      ELSE  %bug entire line segment as 1 char%
         tptr[1] _ entry.stccnt;
   RETURN ;
```

```
    END.
(cnvladj) %convert leval adjust string to leval adjust count%
PROCEDURE( ladjstr REF %leval adjust string%);                          1E
    LOCAL count = 0, i, char;
    FOR i _ 1 UP UNTIL > ladjstr.L DO
        CASE char _ *ladjstr*[i] OF
            = 'u: BUMP count;
            = 'd: BUMP DOWN count;
        ENDCASE NULL;
    RETURN(count);
    END.
(cnvvsp) %convert viewspec string to viewspec words%
PROCEDURE( vstring REF %viewspec string%, windowid);                    1F
    %returns a pointr to a block containing:
        two updated viewspec words
        addr of content analyzer program
        addr of sequence generator program
        viewspec string itself %
    LOCAL vspec REF, i,  char, da REF, goodvs;
    LOCAL STRING badvs[20];
    %alocate and initialize viewspec block%
        &da _ dsparea(windowid);
        &vspec _ alobjk(5 + (vstring.L+4)/5); %vs words + vs string%
        vspec _ da.davspec;
        vspec[1] _ da.davspc2;
        vspec[2] _ da.dacacode;  % content analyzer %
        vspec[3] _ da.dausqcod; % sequence generator %
        % put in viewspec string for mouse button entry handling %
            FOR i _ 0 UP UNTIL > (vstring.L+4)/5  DO
                vspec[i+4] _ vstring[i];  %the string itself, brutally%
    %process viewspecs a charcter at a time%
        FOR i _ 1 UP UNTIL > vstring.L DO
            BEGIN
            char _ *vstring*[i];
            vspec _ setlt (char, vspec, vspec[1] : vspec[1], goodvs);
            IF NOT goodvs THEN *badvs* _ *badvs*, char;
            END;
        IF badvs.L THEN NULL %help call to FE%;
    RETURN(makedesc(ublock,&vspec,TRUE));
    END.
FINISH                                                                  2
```

```
< FE, PCPB8-10.NLS.4, >, 4-Nov-77 16:22 ANDY ;;;;
FILE pcpb8 % (arcsubsys,x110,) (arcsubsys,1109,) to (fe,pcpb8,) %
%  Contains interface code for PCP8 structure reading and writing on
the 10. %
% Records %
   (bytes) RECORD                                                    3A
      pcpb81[8], pcpb82[8], pcpb83[8], pcpb84[8];
% constants %
   DECLARE EXTERNAL STRING pcpname = "PCPB8";
   DECLARE EXTERNAL pcpnull = 289; % PCP empty structure %
   DECLARE EXTERNAL CONSTANT
      pcpempty=1, pcpboolean=2, pcpindex=3, pcpinteger=4, pcpbitstr=5,
      pcpcharstr=6, pcplist=7, pcppad=9, pcpany=0;
   DECLARE EXTERNAL CONSTANT
      wlistend=200, wlistreset=201, wlistnop=202,
      rlistreset=103, wlistoverflow=1001,
      rlistignore=100, rlistzone=101, rlistnop=102;
% procedures %
   (rlist) % read a list %                                           5A
      COROUTINE(
         parms REF, % address of PCPB36 structure %
         zone); % free sto zone %
      % This coroutine is opened with an address of a PCP data
      structure. No results on the OPENPORT. Each subsequent PCALL
      acccepts args type, dest, length: either type is zero, or rlist
      should ABORT when next element is not of type TYPE. If element
      type is a list, length should be same as LENGTH or ABORT will be
      generated. If type is not a list, then LENGTH elements will be
      read and stored starting at dest. (Error if not all same type).
      If dest is not zero, the resulting value is stored thru dest. May
      be PCALLed anytime with type=zero being the only argument. %
      LOCAL
         type, length, dest REF, % PCALL args %
         strlen, % structure length %
         ptr, % byte pointer into structure %
         ptype, % element type from data structure %
         integer, % place to store integer %
         i, % index %
         save, % temp %
         value; % element value %
      PORT ENTRY
      ptr _ 4410B8 + &parms.RH;
      EXIT type _ PCALL (: length, &dest);
      LOOP
         BEGIN
         IF type <= rlistignore THEN
            BEGIN
            CASE (ptype _ ^ptr) OF
               =pcplist:
                  BEGIN
                  strlen _ pcpgtl(ptr: ptr);
                  IF type AND type#rlistignore AND
                     ( type#pcplist OR strlen # length ) THEN
                        err($"Bad argument in RLIST");
                  value _ strlen;
                  END;
```

```
        =pcpbitstr: % 32 bits per 36 bit word !! %
           BEGIN
           strlen _ pcpgtl(ptr: ptr);
           value _ getblk( (strlen+63)/32, zone);
           [value] _ strlen;
           strlen _ (strlen+7)/8;
           save _ value+440888+1;
           FOR i _ 1 UP UNTIL > strlen DO
              ^save _ ^ptr;
           END;
        =pcpcharstr:
           % on the 11, store in compacted form if in dpyzone %
           BEGIN
           strlen _ pcpgtl(ptr: ptr);
           value _ getstring(strlen, zone);
           [value + (strlen+4)/5 ] _ 0; % zero last word %
           FOR i_1 UP UNTIL > strlen DO
              *[value]*[i] _ ^ptr;
           [value].L _ strlen;
           END;
        =pcpinteger:
           BEGIN
           integer _ 0;
           integer.pcpb84 _ ^ptr;
           integer.pcpb83 _ ^ptr;
           integer.pcpb82 _ ^ptr;
           integer.pcpb81 _ ^ptr;
           IF integer .A 2810 # 0 THEN integer _ integer .V
           74810;
              % extend sign bit %
           IF type=pcpinteger AND length AND &dest THEN value _
           integer ELSE value _ $integer;
           END;
        =pcpboolean:
           BEGIN
           value _ ^ptr;
           END;
        =pcpempty:
           BEGIN
           value _ 0;
           END;
        =pcpindex:
           BEGIN
           value _ 0;
           value.pcpb82 _ ^ptr;
           value.pcpb81 _ ^ptr;
           END;
        =pcppad:
           BEGIN
           REPEAT CASE;
           END;
        ENDCASE err($"Bad PCP data type in RLIST");
     CASE type OF
        =0: NULL;
        =rlistignore: % ignore n things (incl. lists ) %
           BEGIN
```

```
                &dest _ 0; % force no result storing %
                IF ptype=pcplist THEN
                   length _ length+strlen;
                   % adjust length to skip over list also %
                IF (length _ length-1) > 0 THEN REPEAT LOOP;
                   % do more if not done, else PCALL back below %
                END;
             # ptype: err($"Bad argument -rlist");
             =pcplist: NULL;
             ENDCASE
                BEGIN
                IF length AND &dest THEN
                   BEGIN
                   dest _ value;
                   BUMP &dest;
                   END;
                IF (length_length-1) > 0 THEN REPEAT LOOP;
                END;
          END
       ELSE
          CASE type OF
             =rlistzone: zone _ length; % set zone %
             =rlistreset: ptr _ length; % re-establish position %
             =rlistnop:
                BEGIN % read type without advancing %
                save _ ptr;
                ptype _ ^ptr;
                ptr _ save;
                END;
             ENDCASE err($"BAD arg -rlist");
       type _ PCALL (ptype, value, ptr: length, &dest);
       END;
    END.
(wlist) % write a PCPB36 data list %                                    5B
    COROUTINE(
       parms REF, % address to write it in %
       n); % number of WORDS available in that block %
    % generate a PCP data structure one element at a time. Each PCALL
    builds one element.IF the type is pcplist, zero value means
    unknown length, else actual length is checked, and sub-elements
    will be build with subsequent PCALLs. If the type is pcpcharstr,
    the value is an a-string address and it will be copied into the
    data structure. %
    % ** later change for more efficient stack usage ** %
    LOCAL CONSTANT listlevmax=10;
    LOCAL
       ptr, ptr2, % byte pointer into structure %
       listlev _0, % nesting level indicator %
       address1 _ &parms, % first word %
       type, value, % element type and value %
       wdlen, % string length in words %
       i, % index %
       listck[listlevmax], % expected list length %
       listele[listlevmax], % element count %
       listpos[listlevmax]; % place to store real count %
    PORT ENTRY
```

```
    ptr _ 4410B8 + &parms.RH;
    EXIT type _ PCALL (: value);
    LOOP
       BEGIN
       IF type < wlistend THEN
          BEGIN
          ^ptr _ type;
          CASE type OF
             =pcpcharstr:
                BEGIN
                wdlen _ ([value].L + 3)/4; % words needed -1 %
                IF ptr.RH-address1 + wdlen >= n THEN
                   ptr.RH _ ptr.RH -
                      (address1 := wlhelp(address1, wdlen+2: n)) +
                      address1;
                ptr _ pcpwrl([value].L, ptr); % write length %
                FOR i_1 UP UNTIL > [value].L DO
                   ^ptr _ *[value]*[i];
                END;
             =pcplist:
                BEGIN
                BUMP listlev; % count nesting %
                IF listlev >= listlevmax THEN err($"too deep
                -wlist");
                listpos[listlev] _ ptr;
                listele[listlev] _ -1; % element count %
                   % it will get bumped at end of case %
                listck[listlev] _ value; % to check at end %
                ptr _ pcpwrl(0, ptr);
                END;
             =pcpindex:
                BEGIN
                ^ptr _ value.pcpb82;
                ^ptr _ value.pcpb81;
                END;
             =pcpinteger:
                BEGIN
                value _ [value];
                ^ptr _ value.pcpb84;
                ^ptr _ value.pcpb83;
                ^ptr _ value.pcpb82;
                ^ptr _ value.pcpb81;
                END;
             =pcpempty: NULL;
             =pcpboolean:
                ^ptr _ IF value THEN 1 ELSE 0;
             =pcpbitstr: % 32 bits per 36 bit word %
                BEGIN
                wdlen _ ([value]+31)/32; % words needed %
                IF ptr.RH-address1 + wdlen >= n THEN
                   ptr.RH _ ptr.RH -
                      (address1 := wlhelp(address1, wdlen+2: n)) +
                      address1;
                ptr _ pcpwrl([value], ptr); % write length %
                ptr2 _ value+440888+1;
                wdlen _ ([value]+7)/8;
```

```
            FOR i_1 UP UNTIL > wdlen DO
               ^ptr _ ^ptr2 ;
            END;
         =pcppad: NULL;
         ENDCASE err($"Bad pcp type -WLIST");
      BUMP listele[listlev];
      END
   ELSE
      CASE type OF
         =wlistend: % signals end of list %
            BEGIN
            CASE listck[listlev] OF
               =0, =listele[listlev]: NULL;
               ENDCASE err($"list err -wlist");
            pcpwrl(listele[listlev], listpos[listlev]);
               % write length %
            BUMP DOWN listlev;
            IF listlev < 0 THEN err($"nesting err -wlist");
            BUMP listele[listlev]; % count the (list) element now
            %
               % it will not get bumped at end %
            END;
         =wlistreset: % reset writing position %
            BEGIN
            ptr _ value;
            listlev _ 0;
            listele[listlev] _ 0;
            END;
         =wlistnop: % do nothing % NULL;
         ENDCASE err($"Bad type -WLIST");
   IF ptr.RH-address1 > n THEN
      ptr.RH _ ptr.RH -
         (address1 := wlhelp(address1, 1: n)) + address1;
   type _
      PCALL ((ptr.RH-address1)*4+4-ptr.bpbitpos/8, ptr: value);
      % number of 8-bit BYTES, pointer %
   END;
END.
(wlhelp) % obtain help for area overflow %                              5C
   PROCEDURE(
      address1, % first address of area %
      needed); % number of words needed %
   % try to get somebody to allocate and copy so we can go on %
   IF HELP(wlistoverflow, address1, needed: address1, needed) =
   gothelp THEN RETURN(address1, needed)
   ELSE err($"area overflow -wlist");
   END.
(pcpgtl) % get length from structure %                                  5D
   PROCEDURE(ptr); % pointer %
   LOCAL len_0;
   len.pcpb82 _ ^ptr;
   len.pcpb81 _ ^ptr;
   RETURN(len, ptr);
   END.
(pcpwrl) % write PCP length field %                                     5E
   PROCEDURE(
```

```
        len, % length value %
        ptr); % pointer to length position %
    ^ptr _ len.pcpb82;
    ^ptr _ len.pcpb81;
    RETURN(ptr);
    END.
FINISH
```

(resetvs)	<nine, psedt1, 0661>	CATCHPHRASE	5G3C10
(rplproc)	<nine, psedt1, 03730>	LOCAL	5W2R
(sigload)	<nine, psedt1, 06952>	CATCHPHRASE	5AB3E24
(supper)	<nine, psedt1, 01371>	LOCAL	5N2E
(updcsp)	<nine, psedt1, 06910>	PROCEDURE	5V1B
(xappend)	<nine, psedt1, 016>	LOCAL	5A1A
(xbreak)	<nine, psedt1, 0102>	LOCAL	5C1A
(xclear)	<nine, psedt1, 03117>	PROCEDURE	5D1A
(xclose)	<nine, psedt1, 0147>	PROCEDURE	5E1A
(xcmgrp)	<nine, psedt1, 0594>	PROCEDURE	5G2B
(xcmst)	<nine, psedt1, 0568>	PROCEDURE	5G2A
(xconnect)	<nine, psedt1, 05350>	PROCEDURE	5F1A
(xcopy)	<nine, psedt1, 0429>	LOCAL	5G1A
(xcreate)	<nine, psedt1, 0686>	LOCAL	5H1A
(xdelete)	<nine, psedt1, 0737>	LOCAL	5I1A
(xdiropt)	<nine, psedt1, 06593>	LOCAL	5G2C
(xentry)	<nine, psedt1, 03265>	PROCEDURE	5K1
(xexpand)	<nine, psedt1, 04573>	LOCAL	5L1
(xexpunge)	<nine, psedt1, 06805>	PROCEDURE	5M1A
(xforce)	<nine, psedt1, 01244>	LOCAL	5N1A
(xfreeze)	<nine, psedt1, 01416>	LOCAL	5O1A
(xfterm)	<nine, psedt1, 03855>	LOCAL	5P1A
(xgetjumpring)	<nine, psedt1, 04744>	PROCEDURE	5V1E
(xgettext)	<nine, psedt1, 01477>	LOCAL	5Q1A
(xgoto)	<nine, psedt1, 04961>	LOCAL	5R1A
(xhandle)	<nine, psedt1, 03250>	PROCEDURE	5S1A
(xinit)	<nine, psedt1, 04196>	PROCEDURE	5T1
(xinsert)	<nine, psedt1, 01536>	PROCEDURE	5U1A
(xinsstatement)	<nine, psedt1, 01751>	LOCAL	5U1C
(xjmpcnt)	<nine, psedt1, 07677>	PROCEDURE	5V1F
(xjump)	<nine, psedt1, 08156>	PROCEDURE	5V1A
(xjumpaddr)	<nine, psedt1, 03155>	LOCAL	5V1D
(xjumpreturn)	<nine, psedt1, 04836>	PROCEDURE	5V1C
(xload)	<nine, psedt1, 03437>	PROCEDURE	5W1A
(xlogout)	<nine, psedt1, 05671>	PROCEDURE	5X1A
(xmark)	<nine, psedt1, 02412>	LOCAL	5Y1A
(xmove)	<nine, psedt1, 02507>	LOCAL	5A@1A
(xnswinit)	<nine, psedt1, 07750>	PROCEDURE	5T3
(xopen)	<nine, psedt1, 02800>	LOCAL	5AA1A
(xoutput)	<nine, psedt1, 02879>	PROCEDURE	5AB1A
(xquitsubsys)	<nine, psedt1, 03836>	PROCEDURE	5AC1A
(xsubsget)	<nine, psedt1, 05040>	PROCEDURE	5R1B

```
< NINE, PSEDT1.NLS;68, >, 25-Jul-78 02:48 GAS2 ;;;;
FILE psedt1 % (arcsubsys,L109,) (RELNINE,psedt1.rel,) %
%Allow system interface calls.  They exist in the procedures gpget,
gpadjbsz, coutproc, xreset, xshow, csubsta, (1 each, I think) and
cshodskspa (7  occurrences)%
   ALLOW! NOMESS
%compile-time switch%
   SET NSW = FALSE;  %TRUE => compile to (RELNINE,wmpsedt1.rel,) %
% Declarations %
   % The following are global to this file only and instead should be
   ref'ed in each procedure that uses them.  Compile Procedure will not
   work if the procedure uses one of these. %
   REF msgda, rawchr, inpt, tda;
   REF nisloader, symloc, ddtsptr;
% EDITOR SUBSYSTEM %
   %append%
      %append PCP interface routine%
         (xappend) %Execute Append Command%                              5A1A
            PROCEDURE (
               %FORMALS%
                  entity REF,   %entity type%
                  sourceptr REF,        %source pointer%
                  destptr REF,  %destination pointer%
                  litptr REF,   %string to insert between dest & srce%
                  deleteflag    %delete the appended stmt?%
                  );
               LOCAL adstr[40], source REF, destination REF, literal
               REF, retry REF;
            %----------------------%
            CASE ELEM #entity#[cwtype] OF
               = 29 %- statement -%:
                  BEGIN
                  IF &sourceptr THEN &source _ ELEM
                  #sourceptr#[tppair];
                  IF &destptr THEN &destination _ ELEM
                  #destptr#[tppair];
                  IF &litptr THEN &literal _ ELEM #litptr#[tppair];
                  IF NOT nortnrings THEN
                     clist(ctlcfm, destination.stfile, source.stfile);
                  IF NOT nodisplay THEN
                     dpset(dsprfst, destination, source, endfil);
                  FIND SE(destination) ^curmkr;
                  IF NOT source.stastr THEN
                     cappsta(destination, source, &literal,
                     &literal+d2sel, deleteflag)
                  ELSE
                     capptex(destination, &source, &source+d2sel,
                     &literal, &literal+d2sel);
                  IF NOT nortnrings THEN clupdt();
                  END;
               ENDCASE
               IF NOT donthelp THEN
                  BEGIN
                  &retry _ HELP (badarg, $"The selection type was
                  incorrect.  The only allowed type is Statement.",
                  cindex);
```

```
            source _ ELEM #retry#[tppair];
            REPEAT CASE (ELEM #retry#[cwtype]);
            END
         ELSE ABORT (badarg, $"The selection type was
         incorrect.");
      RETURN;
      END.
   %append core routines%
      (cappsta)                %Core NLS Append Statement Command% 5A2A
         PROCEDURE (tostid, fromstid, tp1, tp2, deleteflag);
         LOCAL srsub, tosub, level;
         REF tp1, tp2;
           %msnst();%
         IF tostid = fromstid THEN ABORT(badappend,$"illegal
         append");
         IF (srsub _ getsub(fromstid)) # fromstid THEN
            BEGIN
            movtst( tostid, srsub, getail(srsub));
            IF (tosub _ getsub(tostid)) # tostid THEN
               BEGIN
               tosub _ getail(tosub);
               level _ levsuc;
               END
            ELSE level _ levdown;
            cmovgro( tosub, level, srsub, getail(srsub), FALSE, 0);
            END;
         ST tostid _ SF(tostid) SE(tostid), tp1 tp2, SF(fromstid)
         SE(fromstid);
         IF deleteflag THEN cdelgro(fromstid, fromstid, 0);
           %msfst();%
         RETURN;
         END.

      (capptex)                %Core NLS Append Text Command%      5A2B
         PROCEDURE (stid, t1, t2, l1, l2);
         REF t1, t2, l1, l2;
         %msnst();%
         ST stid _ SF(stid) SE(stid), l1 l2, t1 t2;
         %msfst();%
         RETURN;
         END.

%archive%              %moved to PSEDIT-EXTRA%
%break%
   %break PCP interface routine%
      (xbreak) %Execute Break Command%                              5C1A
         PROCEDURE (
            %FORMALS%
               entity REF,  %entity type%
               sourceptr REF,  %source pointer%
               destptr REF,  %destination pointer%
               level,        %level relative adjustment count%
               option REF,   %text to insert at front of new
               statement or "centered"/NULL for break window%
               stype REF,  %type of window split (horiz/vert)%
               tprtn         %if TRUE, return text pointer(s)%
```

```
        );
    LOCAL STRING locstr[5];
    LOCAL centered _ FALSE, direction, destination REF,
    daold, danew, literal REF;
    LOCAL LIST tplist[4];
  %----------------%
  CASE ELEM #entity#[cwtype] OF
    = 29:  % statement %
      BEGIN
      IF &destptr THEN &destination _ ELEM
      #destptr#[tppair];
      IF &option THEN &literal _ ELEM #option#[tppair]
      ELSE &literal _ 0;
      IF NOT nortnrings THEN
        clist(ctcmk, destination.stfile, endfil);
      IF NOT nodisplay THEN
        dpset(dsprfst, destination, endfil, endfil);
      curmkr _ cbresta(&destination, level, &literal,
      &literal + d2sel);
      curmkr[1] _ 1; % to start of broken stmt %
      IF NOT nortnrings THEN clupdt();
      %
      IF tprtn THEN
        BEGIN
        #tplist# _ destination, 0, curmkr, 0;
        RETURN (tplist);
        END
      ELSE
        RETURN (empty);
      %
      END;
    = 33:  % window %
      BEGIN
      dpset(dspno, endfil, endfil, endfil);
      IF &option THEN
        CASE ELEM #option#[cwtype] OF
          =  54:  % centered %
            centered _ TRUE;
        ENDCASE;
      IF &stype THEN
        CASE ELEM #stype#[cwtype] OF
        = 51:  % horizontal break %
          direction _ horizontal;
        = 52:  % vertical break %
          direction _ vertical;
        ENDCASE
          direction _ vertical
      ELSE
        ABORT(badarg, $"The direction was incorrect.");
      daold _ wbreak(&sourceptr, centered, direction,
      &destptr: danew);
      % Update screen for new windows %
        dafrmt (daold, 0);
        dafrmt (danew, 0);
      END;
  ENDCASE
```

```
            ABORT(badarg, $"The selection type was incorrect.");
        RETURN;
        END.

    %break core routines for break statement.  break window routines
    are in file frontend%
        (cbresta)                %Core NLS Break Statement Command%  5C2A
          PROCEDURE ( bug, rlevcnt, tp1, tp2);
          LOCAL newstd; %new stid%
          REF bug, tp1, tp2;
            %msnst();%
          newstd _ newrng(bug.stfile);
          insgrp(bug, rlevcnt, newstd, newstd);
          FIND bug > CH $PT ^p1 $NP ^p2;
          cldtxt($p1, $p2);
          IF &tp1 THEN ST newstd _ tp1 tp2, p2 SE(bug)
          ELSE ST newstd _ p2 SE(bug);
          ST bug _ SF(bug) p1;
            %msfst();%
          RETURN(newstd);
          END.

%clear%
    %clear PCP interface routine%
        (xclear) PROCEDURE;                                          5D1A
        RETURN;
        END.
%close%
    %close PCP interface routine%
        (xclose) PROCEDURE (fileno);                                 5E1A

        %this statement probably not necssary
        [flntadr(fileno)].flnoclose _ FALSE;
        %
        close (fileno);
        RETURN;
        END.
%connect%
    %connect PCP interface routine%
        (xconnect)   % CL: ;  Execute Connect Command %
        PROCEDURE (entptr REF LIST, destptr REF LIST, paramptr REF
        LIST, rtnlist REF LIST);                                     5F1A
            % Procedure description
              FUNCTION
                 X-routine for connect.  Currently supports only the
                 connect to Directory command.
              ARGUMENTS
                 entptr--REF-list addr for block for command word.
                 destptr--REF-list addr for block for
                 LSEL(#"DIRECTORY")
                 paramptr--REF-list addr for block for
                 LSEL(#"PASSWORD")
                 rtnlist--REF-list addr for x-routine results.
              RESULTS
                 proc-value
              NON-STANDARD CONTROL
```

```
        none
      GLOBALS
        none
      %
    % Declarations %
      LOCAL
        entity REF, destination REF, parameters REF,
        console, tp REF, dskcn1, dskcn2, dir1, dir2;
      LOCAL STRING
        locstr[50], erstng[200], send[10];
    % decode parameters %
      &destination _ ELEM #destptr# [tppair];
      &parameters _ ELEM #paramptr# [tppair];
    % process by entity type %
      CASE ELEM #entptr# [cwtype] OF
        %
        = 177 %%- display -%%:
          BEGIN
          &tp _ &destination+d2sel;
          *locstr* _ destination tp;
          console _ VALUE($locstr, 8);
          %%suppress the current display%%
            shutdis();
            CASE nldevice OF %%turn tty simulation off%%
              = devlproc: NULL;
              = imlac0, = imlac1:
                BEGIN
                *send* _ begmsg, 1+remfudge, remtsf;
                !sout(dspjfn, chbmty+$send, -send.L);
                END;
               ENDCASE %%tasker%%
                err($"No Tasker");
          xconterm(console, IF parameters = 1 THEN TRUE ELSE
          FALSE  %%input flag%%);
          LOOP inpt();
          END;
        = 47 %%- tty -%%:
          BEGIN
          &tp _ &destination+d2sel;
          *locstr* _ destination tp;
          console _ VALUE($locstr, 8);
          xconterm(console, IF parameters = 1 THEN TRUE ELSE
          FALSE %%input flag%%);
          IF parameters = $input THEN
            LOOP inpt();
          END;
        %
        = 9 %- directory -%:
          BEGIN
          IF parameters = 0 THEN
            BEGIN  % no password entered, ensure valid text
            pointers anyway %
            FIND SF(*locstr*) ^parameters;
            parameters[ d2sel] _ parameters;
            parameters[ d2sel+1] _ parameters[1];
            END;
```

```
                !gjinf();
                dir1 _ R2;
                !gtdal(IF tops20flag THEN -1 ELSE 0);
                dskcn1 _ R2 - R1;
                cconfildir(&destination, &destination+d2sel,
                &parameters, &parameters+d2sel);
                !gjinf();
                dir2 _ R2;
                !gtdal(IF tops20flag THEN -1 ELSE 0);
                dskcn2 _ R2 - R1;
                IF dskcn1 > 0 THEN
                   BEGIN
                   gdname( dir1, $locstr);
                   *erstng* _
                      EOL, *locstr*, " OVER ALLOCATION BY ",
                      STRING(dskcn1), " PAGES!";
                   END;
                IF (dskcn2 > 0) AND (dir1 # dir2) THEN
                   BEGIN
                   gdname( dir2, $locstr);
                   *erstng* _
                      *erstng*, EOL, *locstr*, " OVER ALLOCATION
                      BY ", STRING(dskcn2), " PAGES!";
                   END;
                IF erstng.L THEN #rtnlist#[1] _ *erstng*;
                END;
             ENDCASE ABORT (badarg, $"The type argument was
             incorrect.");
          % Return %
             RETURN;
          END.

   %connect core routines%
      (cconfildir)   % GB: core NLS Connect to File Directory
      procedure %
      PROCEDURE (dir1, dir2, pass1, pass2 % => no value %);      5F2A
         % Procedure description
            FUNCTION
               Connects to the directory specified, checking that
               the password matches the password specified.  The
               directory and password may be upper or lower case.
            ARGUMENTS
               dir1 - TEXT POINTER - start of directory name
               dir2 - TEXT POINTER - end of directory name
               pass1 - TEXT POINTER - start of password
               pass2 - TEXT POINTER - end of password
            RESULTS
               none
            NON-STANDARD CONTROL
               Error if the directory does not exist or the password
               is incorrect.
            GLOBALS
               CHANGED
                  jpassw, ipassw
               READ ONLY
                  chbmty
```

```
    EXAMPLE
       cconfildir($tp1, $tp2, $tp3, $tp4)
    %
  % Declarations %
    LOCAL
       dirnum,    % holds directory number %
       bytptr,    % byte pointer for use in STDIR and CNDIR
       jsies %
       recog;     % flag to do recognition or not %
    LOCAL STRING
       dirname[40],     % string that gets ASCIZ directory
       name %
       password[40];    % string that gets ASCIZ password
       name %
    REF dir1, dir2, pass1, pass2;
  % build needed upper case ASCIZ strings for jsies %
    FIND dir1 > $(SP/TAB) ^dir1 dir2 < $(SP/TAB) ^dir2;
    *dirname* _ + dir1 dir2, 0;
    FIND pass1 > $(SP/TAB) ^pass1 pass2 < $(SP/TAB) ^pass2;
    *password* _ + pass1 pass2, 0;
  % perform recognition if last char was a ^F, <ALT>, or EOL
  %
    CASE *dirname*[dirname.L - 1] OF
       =$ctlf, =$ascalt, =EOL:
          BEGIN
          *dirname*[dirname.L - 1] _ 0;      % null last
          byte %
          recog _ -1;         % turn on recognition %
          END;
       ENDCASE recog _ 0;     % no recognition %
  % contruct byte pointer to directory name for STDIR %
    bytptr _ chbmty + $dirname;
  % get directory number given directory name; no recognition
  %
    !stdir( recog, bytptr, 0);
       GOTO nodir;
       GOTO nodir;   % ambiguous and no match generate error
       %
       GOTO gotdir;  % got a good number now %
  % ambiguous or illegal directory name passed %
    (nodir): err($"no such directory");                    5F2A7A
  % have a valid directory number by now %
    (gotdir): dirnum _ R1.RH;                              5F2A8A
  % contruct byte pointer to password name for STDIR %
    bytptr _ chbmty + $password;
  % now connect; only error possible since we are already
  logged in and we got a valid directory number above, is
  invalid password %
    IF NOT SKIP !cndir( dirnum, bytptr ) THEN
    err($"incorrect password");
  % success, so save password incase we call conjdir or
  conidir %
    *jpassw* _ *password*;
    *ipassw* _ *password*;
  % all done so return %
    RETURN;
```

BLP, 16-Aug-78 00:18 < NINE, PSEDT1.NLS;68, > 8

END. %%

```
%copy%
   %copy PCP interface routine%
      ( xcopy) %Execute Copy Command%                                    5C1A
         PROCEDURE (
            %FORMALS%
               stype REF,              %source type%
               sourceptr REF,          %source pointer%
               dtype REF,              %destination type%
               destptr REF,            %destination pointer%
               level,                  %level adjustment count%
               vs REF,                 %viewspec words%
               txtrtn,        %if TRUE, return text of statement%
               tprtn,         %if TRUE, return text pointer(s)%
               reslist REF LIST        %result lst%
               );
            LOCAL
               i,    %counter in FOR loop%
               savedest,  %to save value when returning text%
               retry REF,    %for HELP return%
               tlngth, endstid, grplist REF,
               rhostn, rhost2, adstr[40],
               source REF, destination REF,
               sttype,
               % stuff for copy directory %
                  info,      % record saying what was requested %
                  gropk,     % record saying how to group things %
                  sortk;     % record saying how to sort things %
            LOCAL TEXT POINTER f1, f2;
            LOCAL STRING filstr[200], locfil[200], badmsg[200],
            result[2000];
            LOCAL LIST tplist [8], txtlist[1];
         %----------------------%
         &source _ IF &sourceptr THEN ELEM #sourceptr#[tppair] ELSE
         0;
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         % initialize result list %
            #reslist#[1] _ ;
         CASE (sttype _ ELEM #stype#[cwtype]) OF
            = 30 %- link -%:
               BEGIN
               lnkprs( &source, $adstr);
               source _ adstr[ls];
                  source[1] _ adstr[ls+1];
               [&source+d2sel] _ adstr[le];
                  [&source+d2sel+1] _ adstr[le+1];
               END;
            ENDCASE;
         CASE sttype OF
            %text/structure entities%
               = 2 %- character -%, = 11 %- invisible -%, = 1 %-
               text -%:
                  BEGIN
                  IF NOT nortnrings THEN
                     clist(ctcmk, destination.stfile,
                     source.stfile);
```

```
        IF NOT nodisplay THEN
           dpset(dsprfmt, destination, endfil,
           destination);
        curmkr _ destination;
        curmkr[1] _ destination[d2sel+1] + source[d2sel+1]
        - source[1] - 1;
        ccoptex(&destination+d2sel, &source,
        &source+d2sel, FALSE);
        destination[d2sel] _ destination[d2sel] + 1;
        %For returning handles to ISI%
        IF NOT nortnrings THEN clupdt();
        IF txtrtn THEN #txtlist# _ curmkr;
        END;
     = 3 %- word -%, = 4 %- visible -%, = 8 %- integer -%,
     = 30 %- link -%:
        %note that "number" treated as "integer" %
        BEGIN
        IF NOT nortnrings THEN
           clist(ctcmk, destination.stfile,
           source.stfile);
        IF NOT nodisplay THEN
           dpset(dsprfmt, destination, endfil,
           destination);
        curmkr _ destination;
        curmkr[1] _ destination[d2sel+1] + source[d2sel+1]
        - source[1];
        ccoptex(&destination+d2sel, &source,
        &source+d2sel, TRUE);
        destination[d2sel] _ destination[d2sel] + 1;
        %For returning handles to ISI%
        IF NOT nortnrings THEN clupdt();
        IF txtrtn THEN #txtlist# _ curmkr;
        END;
     %note that "number" treated as "integer" %
     % note that "number" treated as "integer"%
     = 29 %- statement -%:
        BEGIN
        curmkr _ xcmst(&destination, level, &source,
        copyflag, &vs);
        curmkr[1] _ 1;
        IF txtrtn THEN #txtlist# _ curmkr;
        destination[d2sel] _ curmkr;  %For returning
        handles to ISI%
        destination[d2sel+1] _ 1;
        END;
     = 27 %- group -%, = 28 %- plex -%, = 26 %- branch -%:
        BEGIN
        curmkr _ xcmgrp(&destination, level, &source,
        copyflag, &vs);
        curmkr[1] _ 1;
        destination[d2sel+1] _ 0;
        CASE level OF
           = 0: destination[d2sel] _ getsuc(destination);
           = -1: destination[d2sel] _
           getsub(destination);
        ENDCASE
```

```
            BEGIN
            FOR i _ 1 UP UNTIL > level DO
               destination[d2sel] _ getup(destination);
            destination[d2sel] _ getsuc(destination);
            END;
         IF txtrtn THEN
            BEGIN
            &grplist _ alloclist(20);
            endstid _ getend(curmkr);
            grplist _ savedest _ destination[d2sel];
            DO
               BEGIN
               #grplist# !_ destination[d2sel] _
               getnxt(destination[d2sel]);
               IF grplist.L = grplist.M THEN
                  &grplist _ alloclist(grplist.M + 20);
               END
            UNTIL destination[d2sel] = endstid;
            destination[d2sel] _ savedest;
            END;
         END;
      = 52 %- sequential -%:
         BEGIN
         IF NOT nodisplay THEN dpset(dsprfst, destination,
         endfil, dpstp(destination));
         % move file name to local string %
            CASE lnbfls( &source, 0, $filstr) OF
               = lhostn: NULL;
               ENDCASE
                  ABORT(noremote,$"Remote File
                  Manipulations Not Implemented Yet");
            %-NSW%                                          5G1A7A7C2
               % setup text pointers %
                  FIND SF(*filstr*) ^f1 SE(*filstr*) ^f2;
            %-NSW%                                          5G1A7A7C3
            %+NSW%                                          5G1A7A7C4
               %retrieve NSW file%
                  IF NOT wmopen(0,$filstr, 0, 0, $locfil,
                  0) THEN err($"Cannot open sequential
                  file");
               % setup text pointers %
                  FIND SF(*locfil*) ^f1 SE(*locfil*) ^f2;
            %+NSW%                                          5G1A7A7C5
         CASE ELEM #dtype#[cwtype] OF
            = 51 %- one -%: i _ tenfil;
            = 53 %- two -%: i _ heurfil;
            = 54 %- justified -%: i _ justfil;
            = 56 %- assembler -%: i _ assfil;
            = 0: %normal% i _ tenfil;
            ENDCASE ABORT(notyet, $"Not implemented.");
         curmkr _ ccopseqfil (destination, level, $f1, $f2,
         i);
         curmkr[1] _ 1;
         %+NSW%                                              5G1A7A7G
            % delete local copy of sequential file %
               dellocfil($locfil);
```

```
            %+NSW%
            END;
         = cwfile %- file -%:
            BEGIN
            % get and initialize message string %
               *result* _ "Copied Files Are:", CR, LF;
               tlngth _ result.L;
            % parse source file name %
               rhostn _ lnbfls( &source, 0, $locfil);
            % parse destination file name %
               rhost2 _ lnbfls( &destination, 0, $filstr);
            ccopfil(rhostn, $locfil, rhost2, $filstr,
            $result);
            % tell the user what we did %
               IF ( result.L > tlngth ) THEN
                  #reslist#[1] _ *result*
               ELSE #reslist#[1] _ "No Files Copied";
            END;
      %directory entities%
         = 9 %- directory -%:
            BEGIN
            dpset(dspstrc,destination,endfil,endfil);
            info _ gropk _ sortk _ 0;
            *filstr* _ *dirsfname*;
            % convert directory option list %
               IF &dtype THEN
                  FOR i _ 1 UP UNTIL > dtype.L DO
                     cnvarg(&dtype, $tplist, i);
            xdiropt( &source, $tplist, $info, $gropk, $sortk,
            $rhostn, $filstr);
            curmkr _ ccopdir(&destination, level, info,
               gropk, sortk, rhostn, $filstr);
            curmkr[1] _ 1;
            END;
         %
         = 61 %%- archive -%%:
            BEGIN
            curmkr _ ccoparcdir(&destination, level, &source,
            &source+d2sel, &dtype);
            curmkr[1] _ 1;
            END;
         %
      ENDCASE
      IF NOT donthelp THEN
         BEGIN
         *badmsg* _ "The selection type was incorrect.  The
         allowed values are Branch, Character, Group, Integer,
         ";
         *badmsg* _ *badmsg*, "Invisible, Link, Plex,
         Statement, Text, Visible, Word, or Sequential.";
         &retry _ HELP (badarg, $badmsg, cindex);
         source _ ELEM #retry#[tppair];
         REPEAT CASE (ELEM #retry#[cwtype]);
         END
      ELSE ABORT (badarg, $"The selection type was
      incorrect.");
```

```
        %
        IF tprtn THEN
           BEGIN
           #tplist# _ source, source[1], source[d2sel],
           source[d2sel+1];
           #tplist# !_ destination[d2sel], destination[d2sel+1];
           IF NOT (curmkr = destination[d2sel] AND curmkr[1] =
           destination[d2sel+1]) THEN #tplist# !_ curmkr,
           curmkr[1];
           IF ELEM #tplist#[4] = 1 THEN #tplist#[4] _ 0;
           END;
        IF tprtn THEN
           IF txtrtn THEN RETURN(grplist, tplist)
           ELSE RETURN (empty, tplist)
        ELSE
           IF txtrtn THEN RETURN(grplist, empty)
           ELSE RETURN(empty,empty);
        %
        RETURN;
        END.

    %copy/move support routines%
       (xcmst) PROCEDURE(destination, level, source, type, vs);   5G2A
          %copy or move statement%
          LOCAL clistcalled, newsid, proc;
          LOCAL STRING vsstr[50], levstr[50];
          REF proc, destination, source, vs;
          &proc _ (IF type = copyflag THEN $ccopsta ELSE $cmovsta);
          IF clistcalled _ ( type NOT= copyflag AND NOT source.stastr
          AND source.stfile NOT= destination.stfile ) THEN
             clist(15, source.stfile, nofile);
          IF NOT source.stastr THEN
             BEGIN
             newsid _ proc(destination, level, source, &vs);
             IF NOT nodisplay THEN
                IF type = copyflag THEN
                   dpset(dspstrc, newsid, endfil, getnxt(newsid))
                ELSE
                   dpset(dspstrc, source, destination, endfil);
             END
          ELSE %do an insert%
             BEGIN
             newsid _ cinssta(destination, level, &source,
             &source+d2sel);
             IF NOT nodisplay THEN
                dpset(dspstrc, newsid, endfil, getnxt(newsid));
             END;
          IF clistcalled THEN clupdt ();
          RETURN(newsid);
          END.

       (xcmgrp) PROCEDURE(destination, level, source, type, vs);  5G2B
          %copy or move group%
          LOCAL clistcalled, proc, newsid, stid;
          LOCAL STRING vsstr[50], levstr[50];
          REF proc, destination, source, vs;
```

```
    &proc _ (IF type = copyflag THEN $ccopgro ELSE $cmovgro);
    IF clistcalled _ ( type NOT= copyflag AND NOT source.stastr
    AND source.stfile NOT= destination.stfile) THEN
       clist(15, source.stfile, nofile);
    IF NOT source.stastr THEN
       BEGIN
       stid _ (IF type = copyflag THEN dpstp(destination) ELSE
       endfil);
       newsid _ proc(destination, level, source,
       [&source+d2sel], &vs);
       IF NOT nodisplay THEN
          dpset(dspstrc, newsid, IF type = copyflag THEN endfil
          ELSE source, stid);
       END
    ELSE %do an insert%
       BEGIN
       newsid _ cinssta(destination, level, &source,
       &source+d2sel);
       IF NOT nodisplay THEN
          dpset(dspstrc, newsid, endfil, getnxt(newsid));
       END;
    IF clistcalled THEN clupdt ();
    RETURN(newsid);
    END.

(xdiropt)      % parse input for directory commands %      5C2C
    PROCEDURE
       (source REF,   % ptr to txt ptr of dir string %
       dent REF,        % addr of to parameter list %
       info REF,        % adr: record to get request info %
       gropk REF,       % adr: record to get request info %
       sortk REF,       % adr: record to get request info %
       rhostn REF,      % adr: cell to get host number %
       deffil REF       % adr: default file string %
       );
    LOCAL
       elem1, elem2, elem3, elem4, % temp ptrs %
       tptr REF, dentelem REF, i,
       count    % temp counter %
       ;
    LOCAL TEXT POINTER tptr;
    LOCAL STRING tstrng[40];
    LOCAL LIST optlist[4];

    % set up file group string %
       rhostn _ lhostn;
       IF &source THEN
          BEGIN
          tptr _ [&source+d2sel];
          tptr[1] _ [&source+d2sel+1];
          IF FIND tptr < "<ESC> ^tptr THEN
             *deffil* _ "<, source tptr, "<^F>, *deffil*
          ELSE
             IF FIND tptr < "<^F> THEN
                *deffil* _ "<, source tptr, *deffil*
             ELSE
```

```
                *deffil* _ "<, source tptr, ">, *deffil*;
            END;
        % parse the parameter input %
          IF &dent THEN
            FOR count _ 1 UP UNTIL > dent.L DO
              BEGIN
              % convert options lsts via NLS arg cnvrter %
                &dentelem _ ELEM #dent# [count];
                FOR i _ 1 UP UNTIL > dentelem.L DO
                  cnvarg(&dentelem, $optlist, i);
                elem1 _ ELEM #optlist#[1];
                elem2 _ ELEM #optlist#[2];
                elem3 _ ELEM #optlist#[3];
                elem4 _ ELEM #optlist#[4];
              CASE ELEM #[elem1]# [cwtype] OF
                = 68 %- both -%: info.dlidlt _ 2;
                = 50 %- delete -%: info.dlidlt _ 1;
                = 69 %- undelete -%: info.dlidlt _ 0;
                = 70 %- for -%:
                  BEGIN
                  % element 2 must be pointer to a text ptr
                  pair, ig arg cnvtr works right %
                  &tpptr _ ELEM #[elem2]# [tppair];
                  rhostn _ lnbfls( &tpptr, 0, &deffil);
                  IF NOT FIND SF(*deffil*) "< THEN
                    IF &source THEN
                      BEGIN
                      tptr _ [&source+d2sel];
                      tptr[1] _ [&source+d2sel+1];
                      IF FIND tptr < "<ESC> ^tptr THEN
                        *deffil* _
                          "<, source tptr, "<^F>, *deffil*
                      ELSE
                        IF FIND tptr < "<^F> THEN
                          *deffil* _
                            "<, source tptr, *deffil*
                        ELSE
                          *deffil* _
                            "<, source tptr, ">,
                            *deffil*;
                      END;
                  END;
                = 61 %- archive -%:
                  CASE ELEM #[elem2]# [cwtype] OF
                    = 52 %- status -%: info.dliars _ TRUE;
                    = 72 %- tape -%: info.dliart _ TRUE;
                    ENDCASE;
                = 73 %- account -%: info.dliacc _ TRUE;
                = 74 %- no -%:
                  CASE ELEM #[elem2]# [cwtype] OF
                    = 75 %- versions -%: info.dlinvr _ 1;
                    = 76 %- extension -%: info.dlinex _ 1;
                    ENDCASE;
                = 59 %- date -%:
                  CASE ELEM #[elem2]# [cwtype] OF
                    = 61 %- archive -%: info.dlitar _ 1;
```

```
                    = 78 %- creation -%: info.dlitcr _ 1;
                    = 51 %- last -%: info.dlitdm _ 1;
                    = 53 %- first -%: info.dlitov _ 1;
                    = 81 %- read -%: info.dlitrd _ 1;
                    = 82 %- write -%: info.dlitwr _ 1;
                    ENDCASE;
                = 83 %- dump -%: info.dlidmt _ TRUE;
                = 84 %- everything -%:
                  BEGIN
                  info.dliacc _ TRUE;
                  info.dliars _ TRUE;
                  info.dliart _ TRUE;
                  info.dlidmt _ TRUE;
                  info.dlidfr _ TRUE;
                  info.dlilwr _ TRUE;
                  info.dlibyt _ TRUE;
                  info.dlimis _ TRUE;
                  info.dlinrw _ TRUE;
                  info.dliprt _ TRUE;
                  info.dlisiz _ TRUE;
                  info.dlitar _ 2;
                  info.dlitcr _ 2;
                  info.dlitdm _ 2;
                  info.dlitov _ 2;
                  info.dlitrd _ 2;
                  info.dlitwr _ 2;
                  END;
                = 51 %- last -%: info.dlilwr _ TRUE;
                = 85 %- length -%: info.dlibyt _ TRUE;
                = 86 %- miscellaneous -%: info.dlimis _ TRUE;
                = 8 %- number -%:
                  CASE ELEM #[elem2]# [cwtype] OF
                    = 75 %- versions -%: info.dlidfr _ TRUE;
                    = 87 %- accesses -%: info.dlinrw _ TRUE;
                    ENDCASE;
                = 88 %- protect -%: info.dliprt _ TRUE;
                = 55 %- size -%: info.dlisiz _ TRUE;
                = 58 %- time -%:
                  CASE ELEM #[elem2]# [cwtype] OF
                    = 61 %- archive -%: info.dlitar _ 2;
                    = 78 %- creation -%: info.dlitcr _ 2;
                    = 51 %- last -%: info.dlitdm _ 2;
                    = 53 %- first -%: info.dlitov _ 2;
                    = 81 %- read -%: info.dlitrd _ 2;
                    = 82 %- write -%: info.dlitwr _ 2;
                    ENDCASE;
                = 91 %- verbose -%:
                  BEGIN
                  info.dlisiz _ TRUE;
                  info.dlilwr _ TRUE;
                  IF NOT info.dlitwr THEN info.dlitwr _ 1;
                  IF NOT info.dlitrd THEN info.dlitrd _ 1;
                  END;
                = 27 %- group -%:
                  BEGIN
                  gropk _ 0;
```

```
                    IF ELEM #[elem2]# [cwtype] = 77 %- reverse
                    -%  THEN gropk.dlgrvr _ TRUE;
                    CASE ELEM #[elem3]# [cwtype] OF
                       = 73 %- account -%: gropk.dlgacc _ TRUE;
                       = 61 %- archive -%:
                          CASE ELEM #[elem4]# [cwtype] OF
                             = 59 %- date -%: gropk.dlgdar _
                             TRUE;
                             = 52 %- status -%: gropk.dlgars _
                             TRUE;
                             = 72 %- tape -%: gropk.dlgart _
                             TRUE;
                             ENDCASE;
                       = 78 %- creation -%: gropk.dlgdcr _ TRUE;
                       = 50 %- delete -%: gropk.dlgdlt _ TRUE;
                       = 83 %- dump -%:
                          CASE ELEM #[elem4]# [cwtype] OF
                             = 59 %- date -%: gropk.dlgddm _
                             TRUE;
                             = 72 %- tape -%: gropk.dlgdmt _
                             TRUE;
                             ENDCASE;
                       = 53 %- first -%: gropk.dlgdov _ TRUE;
                       = 51 %- last -%: gropk.dlglwr _ TRUE;
                       = 8 %- number -%: gropk.dlgdfr _ TRUE;
                       = 88 %- protect -%: gropk.dlgprt _ TRUE;
                       = 81 %- read -%: gropk.dlgdrd _ TRUE;
                       = 82 %- write -%: gropk.dlgdwr _ TRUE;
                       ENDCASE;
                    END;
                 = 57 %- sort -%:
                    BEGIN
                    sortk _ 0;
                    IF ELEM #[elem2]# [cwtype] = 77 %+ reverse
                    +% THEN sortk.dlsrvr _ TRUE;
                    CASE ELEM #[elem3]# [cwtype] OF
                       = 73 %- account -%: sortk.dlsacc _ TRUE;
                       = 61 %- archive -%:
                          CASE ELEM #[elem4]# [cwtype] OF
                             = 58 %- time -%: sortk.dlstar _
                             TRUE;
                             = 72 %- tape -%: sortk.dlsart _
                             TRUE;
                             ENDCASE;
                       = 93 %- bytesize -%: sortk.dlsbyt _ TRUE;
                       = 78 %- creation -%: sortk.dlstcr _ TRUE;
                       = 50 %- delete -%: sortk.dlsdlt _ TRUE;
                       = 83 %- dump -%:
                          CASE ELEM #[elem4]# [cwtype] OF
                             = 58 %- time -%: sortk.dlstdm _
                             TRUE;
                             = 72 %- tape -%: sortk.dlsdmt _
                             TRUE;
                             ENDCASE;
                       = 51 %- last -%: sortk.dlslwr _ TRUE;
                       = 85 %- length -%: sortk.dlslen _ TRUE;
```

```
                        = 8 %- number -%:
                           CASE ELEM #[elem4]# [cwtype] OF
                              = 87 %- accesses -%: sortk.dlsnac _
                              TRUE;
                              = 81 %- read -%: sortk.dlsnrd _
                              TRUE;
                              = 82 %- write -%: sortk.dlsnwr _
                              TRUE;
                              = 75 %- versions -%: sortk.dlsdfr _
                              TRUE;
                              ENDCASE;
                        = 53 %- first -%: sortk.dlstov _ TRUE;
                        = 81 %- read -%: sortk.dlstrd _ TRUE;
                        = 55 %- size -%: sortk.dlssiz _ TRUE;
                        = 82 %- write -%: sortk.dlstwr _ TRUE;
                        ENDCASE;
                     END;
                  ENDCASE;
               END;
         % done so return %
            RETURN;

         END.

   %copy core routines%
      (ccoparcdir)   % UB: core NLS Copy Archive Directory procedure
      %
      PROCEDURE (stid, rlevcnt, tp1, tp2, params % => statement %);
                                                                  5G3A
         % Procedure description
            FUNCTION
               Not implemented yet.
            ARGUMENTS
               none
            RESULTS
               procedure-value - STID - first new statement
            NON-STANDARD CONTROL
               none
            GLOBALS
               none
            EXAMPLE
               none
            %
         % Declarations %
            REF tp1, tp2;
         err( notyet );
         RETURN (stid); %stid of first new statement%
         END.   %%
```

```
(ccopgro)              %Core NLS Copy Group Command%        5G3B
   PROCEDURE (tostid, rlevcrt, fromstid1, fromstid2, vsptr);
   LOCAL newhead, newtail;  %new stids%
   REF vsptr;
   fromstid1 _ grptst(fromstid1, fromstid2 : fromstid2);
   IF NOT &vsptr THEN
      newhead _ copgrp(tostid, rlevcnt, fromstid1, fromstid2,
      FALSE: newtail)
   ELSE
      newhead _ copfgrp(tostid, rlevcnt, fromstid1, fromstid2,
      &vsptr: newtail);
   RETURN(newhead, newtail);
   END.

(ccopsta)       %***%  %Core NLS Copy Statement Command%   5G3C
   PROCEDURE (tostid, rlevcnt, fromstid, vsptr REF);
   LOCAL
      da REF, savvs1, savvs2,
      sig2, sig3, sig4,         %signal results%
      newsrc; %new stid%
   LOCAL STRING locstr[10];
   IF NOT &vsptr THEN
      BEGIN
      newsrc _ newrng(tostid.stfile);
      insgrp(tostid, rlevcnt, newsrc, newsrc);
      copplist(fromstid, newsrc);
      END
   ELSE
      BEGIN
      % set viewspecs to get only one statement %
         *locstr* _ "eq";
         &da _ lda();
         savvs1 _ da.davspec := vsptr.vs1;
         savvs2 _ da.davspc2 := vsptr.vs2;
         INVOKE (resetvs);
         feedlt( &da, $locstr );
         vsptr.vs1 _ da.davspec := savvs1;
         vsptr.vs2 _ da.davspc2 := savvs2;
         DROP (resetvs);
      newsrc _ copfgrp(tostid, rlevcnt, fromstid, fromstid,
      &vsptr);
      END;
   RETURN(newsrc);

   %Define the catchphrase, resetvs.%

   (resetvs) CATCHPHRASE (:sig2, sig3, sig4);          5G3C10
      BEGIN
      da.davspec _ savvs1;
      da.davspc2 _ savvs2;
      END;

   END.

(ccoptex)              %Core NLS Copy Text Command%         5G3D
```

```
    PROCEDURE (totptr, fromtptr1, fromtptr2, insertspace);
    REF totptr, fromtptr1, fromtptr2;
      %msntx();%
    IF insertspace THEN
       ST totptr _ SF(totptr) totptr, " ",
          $fromtptr1 fromtptr2, %dont move pointers%
          totptr SE(totptr)
    ELSE
       ST totptr _ SF(totptr) totptr,
          $fromtptr1 fromtptr2, %dont move pointers%
          totptr SE(totptr);
      %msftx();%
    RETURN;
    END.

(ccopseqfil)   %Core NLS Copy Sequential File Command%      5G3E
    PROCEDURE (stid, rlevcnt, tp1, tp2, filtyp);
    REF tp1, tp2;
    LOCAL STRING string[100];
    *string* _ tp1 tp2; %file name%
    CASE filtyp OF
       = heurfil: stid _ <SEQFIL, inseqn> (stid, rlevcnt,
       $string, 0);
       = justfil: stid _ <SEQFIL, inseqn> (stid, rlevcnt,
       $string, 1);
       ENDCASE
          stid _ <SEQFIL, inseq> (stid, rlevcnt, $string,
          filtyp);
    RETURN (stid); %stid of first new statement%
    END.

(ccopdir)   % GB: core NLS Copy Directory procedure %
PROCEDURE (stid, rlev, info, gropk, sortk, rhstn, fname % =>
statement %);                                               5G3F
    % Procedure description
       FUNCTION
          Inserts after 'stid' a group of statements containing
          links to the files specified by 'fname'.  A variety
          of constraints can be placed on what is listed, how
          they are grouped, and how they are sorted.  However
          using anything but zero for the third, fourth and
          fifth arguments requires a certain amount of
          sophistication.  See the NLS system definitions of
          these records for more information; FEEDBACK will
          tell you where they are.
       ARGUMENTS
          stid - STID - statement after which to insert the
          links
          rlev - INTEGER - level at which to insert them
             -1 => down one level from 'stid'
             0  => same level as 'stid'
             +n => up n levels from 'stid'
                Global variables levdown (-1), levsuc (0) and
                levup (+1) are recommended for this argument.
          info - DLINFO - record describing what to list, or 0
          gropk - DLGRP - record containing grouping
```

```
            information, or 0
            sortk - DLSORT - record containing sorting
            information, or 0
            rhstn - INTEGER -
               number of the host on which the files reside.
               Only the local host ("lhostn") is currently
               implemented.
            fname - STRING -
               file group name string in TENEX format.  Connected
               directory is assumed if a directory is not
               specified.  Examples:
                  "*.*"
                  "<SMITH>*.PC"
                  "<SMITH>FOO.NLS;3"
         RESULTS
            procedure-value - STID -
               first inserted statement, or "stid" if no
               statements were inserted
         NON-STANDARD CONTROL
            Error if
               host is not the local host
               files are not disk files
               NLS storage problems
               usual TENEX file system errors
         GLOBALS
            READ ONLY
               levup, levdown, levsuc, gtjoif, gtjstr, gtjidl,
               lhostn
         EXAMPLE
            ccopdir(stid, levdown, 0, 0, 0, lhostn,
            $"<smith>*.nls")
               inserts down one level from "stid" a group of
               statements of the form
                  < SMITH, FOO.NLS;7, >
                  < SMITH, BAZ.NLS;52, >  [ Being Modified By
                  NAME (IDENT) ]
                     etc.
         %
      % Declarations %
         LOCAL
            noverf,        % no version numbers flag %
            noextf,        % no extension names flag %
            sortdi,        % sort direction %
            retstid,       % return stid %
            tptr, % temp pointer for building final string %
            chns, % pointer to data structure %
            adlmb,         % address of directory list storage %
            fjfn, % main jfn %
            jfnflgs,       % left half flags for a group jfn %
            flags;         % bits indicating * typed for file name
            fields %
         LOCAL TEXT POINTER
            strbgn, strend, tp1, tp2, tp3;
         LOCAL STRING
            nwname[200],  % current file name link %
            oldname[200], % previous file name link %
```

```
        astr[2000],             % string for listing a file %
        uname[200],             % complete name as entered by
        user %
        udirname[40], % user input directory name %
        ufilname[40], % user input file name %
        uextname[40], % user input extension name %
        uverno[40],             % user input version number %
        ulftovr[40],  % user input beyond version number %
        jfnname[200], % complete name from jfns %
        errstring[200],         % eror message string %
        tstring[200]; % temporary string %
    REF stid, tptr, chns, adlmb, fname;
% initial variables %
    fjfn _ jfnflgs _ flags _ retstid _ &tptr _ &chns _
    &adlmb _ 0;
    sortdi _ sortk.dlsrvr := FALSE;
    noverf _ info.dlinvr := FALSE;
    noextf _ info.dlinex := FALSE;
% find out if remote or local (disk) file %
    CASE rhstn OF
        = lhostn:  NULL;
        ENDCASE err( $"Remote File Manipulations Not
        Implemented Yet" );
% now parse user input strings %
    parseinput( &fname, $udirname, $ufilname, $uextname,
    $uverno, $ulftovr, $uname, $flags);
% get main jfn (old file only, group jfn) %
    IF info.dlidlt THEN
        BEGIN
        IF NOT
            (fjfn _
                sgtjfn(
                    gtjoif .V gtjstr .V gtjidl,
                    $uname, $errstring : jfnflgs
                )
            )
                THEN err( $errstring );
        END
    ELSE
        IF NOT
            (fjfn _
                sgtjfn(
                    gtjoif .V gtjstr, $uname, $errstring :
                    jfnflgs
                )
            )
                THEN err( $errstring );
% now find out if we support this type of file %
    IF NOT chkdev( fjfn ) THEN err( $"only disk files
    supported" );
% get directory list storage %
    IF NOT (&adlmb _ dlgmb(400)) THEN
        err($"Internal Storage Problem");
% cleanup on any problems %
    INVOKE(cdircatch);
% go create the data structure %
```

```
        IF NOT  (&chns _
           dldriver(info, gropk, sortk, tjfn, jfnflgs,
           $errstring, &adlmb)) THEN
              BEGIN
              IF NOT SKIP !rljfn( fjfn ) THEN NULL;
              dlfmb( &adlmb := 0);
              IF errstring.L THEN err($errstring)
              ELSE err($"System Screwup");
              END;
     % broadcast any error message %
        IF errstring.L THEN dismes(2,$errstring);
     % build strings for one file, insert it, & loop for all
     files %
        WHILE &chns DO
           BEGIN
           IF inptrf THEN
              BEGIN
              rstcntlo(); % tell FE to resume output %
              EXIT LOOP;
              END;
           IF sortdi THEN &tptr _ chns.dlgbnu
           ELSE &tptr _ chns.dlgbsc;
           WHILE &tptr DO
              BEGIN
              IF inptrf THEN
                 BEGIN
                 rstcntlo(); % tell FE to resume output %
                 EXIT LOOP 2;
                 END;
              IF NOT tptr.dlfbfl.dlstig THEN
                 BEGIN
                 astr.L _ 0;
                 IF noverf OR noextf THEN
                    BEGIN
                    FIND
                       SE(*[tptr.dlfbal]*) [",] ^tp3 [";/".]
                       ^tp2 [".] ^tp1;
                    IF noverf AND noextf THEN
                       *nwname* _ SF(*[tptr.dlfbal]*) tp1,
                          tp3 SE(*[tptr.dlfbal]*)
                    ELSE
                       IF noverf THEN
                          *nwname* _ SF(*[tptr.dlfbal]*) tp2,
                             tp3 SE(*[tptr.dlfbal]*)
                       ELSE
                          *nwname* _ SF(*[tptr.dlfbal]*) tp1,
                             tp2 SE(*[tptr.dlfbal]*);
                    IF *nwname* = *oldname* THEN GOTO cbypass
                    ELSE *oldname* _ *nwname*;
                    *astr* _ *astr*, *nwname*;
                    END
                 ELSE
                    *astr* _ *astr*, *[tptr.dlfbal]*;
                 % build additional line 1 information string %
                    dlbldai(
                       fjfn, tptr.dlfbpf, tptr.dlfbff,
```

```
                    tptr.dlfbpf, $astr);
                % insert the header %
                    cnvcrlfteol($astr);  %convert CR LF to EOL
                    for files%
                    FIND SF(*astr*) ^strbgn SE(*astr*) ^strend;
                    stid _ cinssta( stid, rlev, $strbgn,
                    $strend);
                    IF NOT retstid THEN retstid _ stid;
                    rlev _ levsuc;
                    astr.L _ 0;
                % build info string for this file & insert it %
                    dlbldi(
                        info, tptr.dlfbpf, tptr.dlfbff,
                        tptr.dlfbpf, $"", $astr);
                    IF astr.L THEN
                        BEGIN
                        cnvcrlfteol($astr);  %convert CR LF to
                        EOL for files%
                        FIND SF(*astr*) ^strbgn SE(*astr*)
                        ^strend;
                        rlev _ levdown;
                        stid _ cinssta( stid, rlev, $strbgn,
                        $strend);
                        rlev _ levup;
                        astr.L _ 0;
                        END;
                    END;
                (cbypass);                                      5G3F12A5D
                IF sortdi THEN &tptr _ tptr.dlfbpb
                ELSE &tptr _ tptr.dlfbnb;
                END;
            &chns _ chns.dlgbnb;
            END;
        % get rid of any lingering storage %
            dlfmb( &adlmb := 0 );
        % make sure we get rid of any jfns %
            IF NOT SKIP !rljfn( fjfn ) THEN NULL;
        % drop catchphrase %
            DROP(cdircatch);
        % return %
            RETURN( IF retstid THEN retstid ELSE stid );
        % catchphrase %
            (cdircatch) CATCHPHRASE();                          5G3F17A
                BEGIN
                CASE SIGNALTYPE OF
                    = notetype :
                    = helptype :
                    = aborttype :
                        BEGIN
                        DISABLE(cdircatch);
                        dlfmb(&adlmb := 0);
                        IF NOT SKIP !rljfn( fjfn ) THEN NULL;
                        END;
                    ENDCASE;
                CONTINUE;
                END;
```

```
END.  %%
```

```
(ccopfil)   % GB: core NLS Copy File procedure %
PROCEDURE (rhost1, file1, rhost2, file2, astr % => no value
%);                                                    5G3C
   % Procedure description
      FUNCTION
         Copies the NLS file(s) specified by 'file1' on host
         'rhost1' to file(s) 'file2' on host 'rhost2'.  Any
         Partial Copies are also copied.  The destination
         files must already exist.  Asterisks (*) in the
         source file specification (file1) must be matched by
         asterisks in the destination file specification
         (file2).  Messages describing the results are put in
         'astr'.  Note: this copies the source file's origin
         statement along with its other statements.
      ARGUMENTS
         rhost1 - INTEGER -
            number of the host on which the files reside.
            Only the local host ('lhostn') is currently
            implemented.
         file1 - STRING -
            file group name in TENEX format.  Connected
            directory is assumed if a directory is not
            specified.  Examples:
               "*.*"
               "<SMITH>*.PC"
               "<SMITH>FOO.NLS;3"
         rhost2 - INTEGER -
            number of the host to which the files are to be
            copied.  Only the local host ('lhostn') is
            currently implemented.
         file2 - STRING -
            file group name in TENEX format.  Connected
            directory is assumed if a directory is not
            specified.  Examples:
               "*.*"
               "<SMITH>*.PC"
               "<SMITH>FOO.NLS;3"
            Note: the *'s here must match one-for-one with the
            *'s in file1.
         astr - STRING - string in which to put messages
      RESULTS
         none
      NON-STANDARD CONTROL
         Error if
            host is not local host
            illegal use of <CTRL-F>, $ or * (see the TENEX
            manual for the correct use of these characters)
      GLOBALS
         gtjoif, gtjstr, lhostn - read only
      EXAMPLE
         ccopfil(lhostn, $"<smith>*.nls", lhostn,
         $"<jones>*.nls", $string)
      %
   % Declarations %
      LOCAL
```

```
        jfn1, % main jfn %
        jfn2, % jfn for destination file %
        jfnfl1,        % left half flags for a group jfn %
        jfnfl2,        % left half flags for destination group
        jfn %
        pcjfn1,        % jfn of partial copy %
        pcjfn2,        % jfn of partial copy %
        flag1,         % bits indicating * typed for file name
        fields %
        flag2;         % * typed for destination file name
        fields %
      LOCAL STRING
        uf1[200],      % complete name as entered by user %
        jfnnm1[200],% complete name from jfns %
        uf2[200],      % complete destination name as entered
        by user %
        udir2[40],     % user input directory destination name
        %
        ufil2[40],     % user input file destination name %
        uext2[40],     % user input extension destination name
        %
        uver2[40],     % user input destination version number
        %
        ulft2[40],     % user destination input beyond version
        number %
        jfnnm2[200],   % complete destination name from jfns %
        errstring[200],        % eror message string %
        tstring[200]; % temporary string %
      REF file1, file2, astr;
    % initial variables %
      jfn1 _ jfnfl1 _ pcjfn1 _ flag1 _ 0;
      jfn2 _ jfnfl2 _ pcjfn2 _ flag2 _ 0;
    % find out if remote or local (disk) file %
      CASE rhost1 OF
        = lhostn:  NULL;
        ENDCASE err( $"remote file manipulations not
        implemented yet" );
      CASE rhost2 OF
        = lhostn:  NULL;
        ENDCASE err( $"remote file manipulations not
        implemented yet" );
    % now parse user source input string %
      parseinput( &file1, $udir2, $ufil2, $uext2, $uver2,
      $ulft2, $uf1, $flag1);
    % parse user destination input string %
      parseinput( &file2, $udir2, $ufil2, $uext2, $uver2,
      $ulft2, $uf2, $flag2);
      % illegal to terminate destination fields with ^F or alt
      %
        CASE *udir2*[udir2.L] OF
          = "<^F>, = "<ESC>: err($"Illegal use of ^F or
          ALTMODE");
          ENDCASE;
        CASE *ufil2*[ufil2.L] OF
          = "<^F>, = "<ESC>: err($"Illegal use of ^F or
          ALTMODE");
```

```
            ENDCASE;
         CASE *uext2*[uext2.L] OF
            = "<^F>, = "<ESC>: err($"Illegal use of ^F or
            ALTMODE");
            ENDCASE;
         CASE *uver2*[uver2.L] OF
            = "<^F>, = "<ESC>: err($"Illegal use of ^F or
            ALTMODE");
            ENDCASE;
         CASE *ulft2*[ulft2.L] OF
            = "<^F>, = "<ESC>: err($"Illegal use of ^F or
            ALTMODE");
            ENDCASE;
      % check for matching astericks %
         IF flag1.dstar AND NOT flag2.dstar THEN err($"Illegal
         use of *");
         IF flag1.fstar AND NOT flag2.fstar THEN err($"Illegal
         use of *");
         IF flag1.estar AND NOT flag2.estar THEN err($"Illegal
         use of *");
         IF flag1.vstar AND NOT flag2.vstar THEN err($"Illegal
         use of *");
      % get main source jfn (old file only, group jfn) %
         IF NOT (jfn1 _ sgtjfn( gtjoif .V gtjstr + 0, $uf1,
         $errstring : jfnfl1 ) )THEN err( $errstring );
      % now find out if we support this type of file %
         IF NOT chkdev( jfn1 ) THEN err( $"only disk files
         supported" );
      % now loop to get all files in the group %
         LOOP
            BEGIN
            % check to make sure this not a PC with * for
            extension name %
               IF NOT chkpcs( jfn1, flag1) THEN GOTO gtcnxtjfn;
            % get actual file name into string for use later
            (maybe) %
               R3 _ 011110B6      %  directory file ext ver %
                  + (IF jfnfl1.lhjb9  THEN 1B6 ELSE 0) %
                  ;Protection %
                  + (IF jfnfl1.lhjb10 THEN 1B5 ELSE 0) % ;Account
                  %
                  + (IF jfnfl1.lhjb11 THEN 4B4 ELSE 0) % ;T %
                  + 1;            % with proper file punctuation
                  %
               jfnflink( jfn1, $jfnnm1, R3);
            % get source PC jfn & find if this user can access
            this PC %
               IF NOT getpcjfn( jfn1, $pcjfn1, $errstring, FALSE)
               THEN
                  BEGIN   % this user can't access this file %
                  *tstring* _ *jfnnm1*, " not copied because [",
                  *errstring*, "];
                  IF flag1 THEN
                     BEGIN
                     *astr* _ *astr*, "   ***   ", *tstring*, CR,
                     LF;
```

```
                    GOTO gtcnxtjfn;
                    END
                ELSE
                    BEGIN
                    IF NOT SKIP !rljfn( jfn1) THEN NULL;
                    err( $tstring );
                    END;
                END;
            % get destination jfns %
              IF NOT jfn2 _
                gtdesjfn( jfn1, pcjfn1, $udir2, $ufil2, $uext2,
                $uver2, $ulft2, flag2, $errstring : pcjfn2)
                THEN
                    BEGIN        % can't copy file (or partial
                    copy) %
                    *tstring* _ *jfnnm1*, " not copied because
                    ", *errstring*;
                    IF flag1 THEN
                        BEGIN
                        *astr* _ *astr*, "   ***   ", *tstring*,
                        CR, LF;
                        GOTO gtcnxtjfn;
                        END
                    ELSE
                        BEGIN
                        IF NOT SKIP !rljfn( jfn1) THEN NULL;
                        IF NOT SKIP !rljfn( pcjfn1) THEN NULL;
                        err( $tstring );
                        END;
                    END;
            % get destination file name into string for use later
            %
               R3 _ 011110B6       %  directory file ext ver %
                  + (IF jfnfl1.lhjb9  THEN 1B6 ELSE 0) %
                  ;Protection %
                  + (IF jfnfl1.lhjb10 THEN 1B5 ELSE 0) % ;Account
                  %
                  + (IF jfnfl1.lhjb11 THEN 4B4 ELSE 0) % ;T %
                  + 1;             % with proper file punctuation
                  %
               jfnflink( jfn2, $jfnnm2, R3);
            % now move the file (and its partial copy) %
              IF NOT copflandpc(jfn1, pcjfn1, jfn2, pcjfn2,
              $errstring) THEN
                  BEGIN   % can't move file (or partial copy) %
                  *tstring* _ *jfnnm1*, " not copied because ",
                  *errstring*;
                  IF flag1 THEN
                      BEGIN
                      *astr* _ *astr*, "   ***   ", *tstring*, CR,
                      LF;
                      GOTO gtcnxtjfn;
                      END
                  ELSE
                      BEGIN
                      IF NOT SKIP !rljfn( jfn1) THEN NULL;
```

```
                    IF NOT SKIP !rljfn( pcjfn1) THEN NULL;
                    err( $tstring );
                    END;
                END;
            % now tell the user we have copied this file %
               *astr* _ *astr*, "   ", *jtnnm1*;
               IF pcjfn1 THEN *astr* _ *astr*, " [and PCJ";
               *astr* _ *astr*, " to ", *jfnnm2*;
               IF pcjfn2 THEN *astr* _ *astr*, " [and PCJ";
               *astr* _ *astr*, CR, LF;
            % now get rid of the jfn for the partial %
               IF NOT SKIP !rljfn( pcjfn1 ) THEN NULL;
            % now get the next file in the group %
               (gtcnxtjfn):                                    5G3G10A10A
               R1.LH _ jfnfl1;
               R1.RH _ jfn1;
               IF NOT SKIP !gnjfn( R1 ) THEN EXIT LOOP;
            END;
        % now get rid of any lingering jfns %
            IF NOT SKIP !rljfn( jfn1 ) THEN NULL;
            IF NOT SKIP !rljfn( pcjfn1 ) THEN NULL;
        % all done now, so return %
           RETURN;
        END.   %%
```

```
%create%
   %create PCP interface routine%
      (xcreate) %Execute Create Command%                              5H1A
         PROCEDURE (
            %FORMALS%
               fnameptr REF, %name of file to create%
               window,       %number of last window bugged%
               rtnlist REF); %return arg%
            LOCAL da REF, rhostn, filename REF;
            LOCAL STRING filstr[200];
         %----------------------%
         IF &fnameptr THEN &filename _ ELEM #fnameptr#[tppair];
         cspupdate _ &da _ dsparea(window);
         % parse input file link %
            rhostn _ lnbfls( &filename, 0, $filstr);
         curmkr _ ccrefil(rhostn, $filstr, window);
            %returns stid to origin%
         curmkr[1] _ 1;
         IF NOT nodisplay THEN dpset( dspyes, curmkr, endfil, endfil
         );
         cspvs _ da.davspec;
         cspvs[1] _ da.davspc2;
         #rtnlist#[1] _ *filstr*;
         RETURN;
         END.

   %create core routines%
      (ccrefil)              %Core NLS Create File Command%          5H2A
         PROCEDURE (rhostn, fname);
         LOCAL stid, jfn, flgbits, errcode, fileno;
         REF fname;
         CASE rhostn OF
            = lhostn:
               BEGIN
               %-NSW%                                                 5H2A4A2
               IF NOT FIND SF(*fname*) ["."] THEN *fname* _ *fname*,
               ".NLS";
               flgbits _ getgtjflg(read, origff, oldvrsn);
               IF NOT (jfn _ sgtjfn(flgbits, &fname, 0 : errcode))
               THEN
                  BEGIN
                  CASE errcode OF
                     =$gjfx24, =$gjfx20, =$gjfx19: NULL;
                        %expect "old file required" or "no such
                        version" (or "no such extension" on TOPS20)%
                     ENDCASE %got something else%
                        BEGIN
                        gtjerr(errcode, $lit, flgbits);
                        ABORT (badcreate, $lit);
                        END;
                  END
               ELSE  %already have a file by that name%
                  ABORT (badcreate, $"A file by that name already
                  exists!");
               %-NSW%                                                 5H2A4A7
```

```
            fileno _ openull($fname);
            stid _ origin;
            stid.stfile _ fileno;
            updtfl(fileno, oldversion, 0);  %update to old%
            RETURN(stid);  %stid of origin%
            END;
         ENDCASE ABORT (noremote, $"Remote File Manipulations Not
         Implemented Yet");
      END.

%delete%
   %delete PCP interface routine%
      (xdelete) %Execute Delete Command%                              5I1A
         PROCEDURE (
            %FORMALS%
               entity REF,              %entity type%
               sourceptr REF,           %source pointer%
               destptr REF,             %destination pointer%
               vs REF,                  %viewspec string%
               window,                  %window number%
               txtrtn,        %if TRUE, return text of statement%
               rtnlist REF              %result list%
               );
            LOCAL
               enttype,
               destination REF, stid, type, da REF, deleteda REF,
               cords, tlngth, rhostn, txtent, dato, adstr[40];
            LOCAL retry, sig2, sig3, sig4;   %signal results%
            LOCAL
               result, endda;
            LOCAL LIST txtlist[1];
            LOCAL TEXT POINTER z1, z2;
            LOCAL STRING filstr[200], badmsg[250], dafilstr[200];
         %----------------------%
         txtent _ FALSE;
         CASE enttype _ ELEM #entity#[cwtype] OF
            = 30 %- link -%:
               BEGIN
               IF &destptr THEN
                  &destination _ ELEM #destptr#[tppair];
               lnkprs( &destination, $adstr);
               destination _ adstr[ls];
                  destination[1] _ adstr[ls+1];
               [&destination+d2sel] _ adstr[le];
                  [&destination+d2sel+1] _ adstr[le+1];
               END;
            = 33 % - window %:
               NULL;
            ENDCASE
               BEGIN
               IF &destptr THEN
                  &destination _ ELEM #destptr#[tppair];
               END;
         CASE enttype OF
            %text and structure entities%
               = 2 %- character -%, = 1 %- text -%, = 11 %-
```

```
            invisible -%:
               BEGIN
               txtent _ TRUE;
               IF NOT nortnrings THEN
                  clist(ctcmk, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dsprfmt, destination, endfil,
                  destination);
               FIND destination ^curmkr;
               cdeltex(&destination, &destination+d2sel, FALSE);
               IF FIND curmkr > ENDCHR THEN FIND curmkr _curmkr;
               IF NOT nortnrings THEN clupdt();
               END;
            = 3 %- word -%, = 4 %- visible -%, = 8 %- integer -%,
            = 30 %- link -%:
               BEGIN
               txtent _ TRUE;
               IF NOT nortnrings THEN
                  clist(ctcmk, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dsprfmt, destination, endfil,
                  destination);
               z1 _ destination[d2sel];
                  z1[1] _ destination[d2sel+1];
               FIND destination ^curmkr;
               IF NOT FIND z1 > SP THEN
                  IF FIND destination < SP ^curmkr THEN NULL;
               cdeltex(&destination, &destination+d2sel, TRUE);
               IF FIND curmkr > ENDCHR THEN FIND curmkr _curmkr;
               IF NOT nortnrings THEN clupdt();
               END;
            = 29 %- statement -%:
               BEGIN
               IF NOT nortnrings THEN
                  clist(ctlcfm, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dspstrc, destination, endfil,
                  dpstp(destination));
               CASE curmkr _ getnxt(destination) OF
                  = destination,
                  = endfil:
                     curmkr _ getbck(destination);
                  ENDCASE;
               curmkr[1] _ 1;
               cdelsta(destination, &vs);
               IF NOT nortnrings THEN clupdt();
               END;
            = 27 %- group -%, = 28 %- plex -%, = 26 %- branch -%:
               BEGIN
               IF NOT nortnrings THEN
                  clist(ctlcfm, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dspstrc, destination, endfil,
                  dpstp(destination));
               CASE curmkr _ getnxt(getend(destination[d2sel]))
               OF
```

```
                    = endfil:
                        curmkr _ getbck(destination);
                    ENDCASE;
                curmkr[1] _ 1;
                cdelgro(destination, destination[d2sel], &vs);
                IF NOT nottnrings THEN clupdt();
                END;
            = 51 %- file -%:
                BEGIN
                cspupdate _ FALSE;
                result _ getstring( 3000, $dspblk);
                *[result]* _ "Deleted Files Are:", CR, LF;
                tlngth _ [result].L;
                rhostn _ lnbfls( &destination, 0, $filstr);
                cdelfil( rhostn, $filstr, result);
                IF ( [result].L <= tlngth ) THEN
                    *[result]* _ "No Files Deleted";
                #rtnlist#[1] _ *[result]*;
                IF FIND SF(*[result]*) "Deleted Files Are:" ^z2
                THEN
                    BEGIN
                    endda _ $dpyarea + dacnt*dal;
                    WHILE (FIND z2 ['<] ^z1 _z1 ['>] ^z2 ) DO
                        BEGIN
                        *filstr* _ z1 z2;
                        FOR &da _ $dpyarea UP dal UNTIL >= endda DO
                            IF da.daexis AND NOT da.daempty THEN
                                BEGIN
                                filnam(da.dacsp.stfile, $dafilstr);
                                IF *filstr* = *dafilstr* THEN
                                    BEGIN
                                    da.daempty _ TRUE;
                                    close(da.dacsp.stfile);
                                    dpset(dspallf, da.dacsp, endfil,
                                    endfil);
                                    END;
                                mkdelfrr(da.dalink, $filstr);
                                END;
                        END;
                    END;
                END;
            = 33;  % window %
                BEGIN
                dpset(dspno, endfil, endfil, endfil);
                % Check for overlappingor non-overlapping windows %
                    IF &destptr THEN
                        BEGIN
                        dato _ wappend(&sourceptr, &destptr);
                        % Update screen for expanded window) %
                            dafrmt(dato, 0);
                        END
                    ELSE wdelete(&sourceptr);  %overlapping%
                END;
            = 50 %- marker -%:
                cdelmar(&destination, &destination+d2sel, lcfile());
            = 54 %- allmarkers -%:
```

```
            cdelallmar(lcfile()); %must know file%
         = 53 %- modifications -%: %to file%      %CHANGE?%
            BEGIN
            stid _ orgstid;
            stid.stfile _ lcfile();
            IF NOT nortnrings THEN
               clist(ctlcfm, stid.stfile, nofile);
            IF NOT nodisplay THEN
               dpset(dspallf, stid, endfil, endfil);
            cdelmodfil(stid.stfile);
            unlkclist (); %check clistt items%
            IF NOT nortnrings THEN clupdt();
            END;
         ENDCASE
         IF NOT donthelp THEN
            BEGIN
            *badmsg* _ "The type argument was incorrect.  The
            allowed values are Allmarkers, Allprograms, Branch,
            Character, Edge, Group, ";
            *badmsg* _ *badmsg*, "Integer, Invisible,
            Lastprogram, Link, Marker, Modifications, Plex,
            Statement, Text, Visible, or Word.";
            retry _ HELP (badarg, $badmsg, cindex);
            REPEAT CASE (retry);
            END
         ELSE ABORT (badarg, $"The type argument was
         incorrect.");
      %
      IF txtrtn AND txtent THEN
         BEGIN
         #txtlist# _ curmkr;
         RETURN($txtlist);
         END
      ELSE
         %
         RETURN;

      %Define the catchphrase, allrefresh%

      (allrefresh) CATCHPHRASE (:sig2, sig3, sig4);          5I1A12
         BEGIN
         DISABLE (allrefresh);
         CASE SIGNALTYPE OF
            = aborttype: alldsp();
         ENDCASE;
         END;

      END.

%delete core routines%
   (cdelallmar)   %Core NLS Delete All Markers Command%       5I2A
      PROCEDURE(fileno);
      LOCAL mk, stid, aring;
      REF aring;
      % make sure we have a locked file %
         stid _ origin;
```

```
        stid.stfile _ fileno;
        lodent( stid, rngtyp : &aring);
        aring.rsub _ aring.rsub;
      mk _ filhdr(fileno) + $mkrtbl - $filhed;
      [mk] _ 0;
      RETURN;
      END.

(cdelfil)   % GB: deletes a group of files and partial copies
%
PROCEDURE (rhstn, fname, astr % => no value %);              5I2B
   % Procedure description
      FUNCTION
         Deletes the NLS file(s) specified by 'fname' on host
         'rhstn'.  Any Partial Copies are also deleted.
         Messages describing the results are put in 'astr'.
      ARGUMENTS
         rhstn - INTEGER -
            number of the host on which the files reside.
            Only the local host ('lhostn') is currently
            implemented.
         fname - STRING -
            file group name string in TENEX format.  Connected
            directory is assumed if a directory is not
            specified.  Examples:
               "*.*"
               "<SMITH>*.PC"
               "<SMITH>FOO.NLS;3"
         astr - STRING - string to receive list of deleted
         files
      RESULTS
         none
      NON-STANDARD CONTROL
         Error if
            host is not local host
            file not disk file
            user not allowed access to file and can't release
            JFN
            can't access or delete file or PC
      GLOBALS
         <lots>
      EXAMPLE
         cdelfil(lhostn, $"*.NLS", $string)
         cdelfil(lhostn, $"<SMITH>FOO.*;*" $string)
      %
   % Declarations %
      LOCAL
         fjfn, % main jfn %
         jfnflgs,      % left half flags for a group jfn %
         pcjfn,        % jfn of partial copy %
         flags;        % bits indicating * typed for file name
         fields %
      LOCAL STRING
         uname[200],           % complete name as entered by
         user %
         udirname[40], % user input directory name %
```

```
        ufilname[40], % user input file name %
        uextname[40], % user input extension name %
        uverno[40],           % user input version number %
        ulftovr[40],  % user input beyond version number %
        jfnname[200], % complete name from jfns %
        errstring[200],       % eror message string %
        tstring[200]; % temporary string %
    REF fname, astr;
% initial variables %
    fjfn _ jfnflgs _ pcjfn _ flags _ 0;
% find out if remote or local (disk) file %
    CASE rhstn OF
        = lhostn:  NULL;
        ENDCASE err( $"remote file manipulations not
        implemented yet" );
% now parse user input strings %
    parseinput( &fname, $udirname, $ufilname, $uextname,
    $uverno, $ulftovr, $uname, $flags);
% get main jfn (old file only, group jfn) %
    IF NOT (fjfn _ sgtjfn( gtjoif .V gtjstr + 0, $uname,
    $errstring : jfnflgs ) )THEN err( $errstring );
% now find out if we support this type of file %
    IF NOT chkdev( fjfn ) THEN err( $"only disk files
    supported" );
% now loop to get all files in the group %
    LOOP
        BEGIN
        % check to make sure this not a PC with * for
        extension name %
            IF NOT chkpcs( fjfn, flags) THEN GOTO getnxtjfn;
        % get actual file name into string for use later
        (maybe) %
            R3 _ 011110B6      %  directory file ext ver %
                + (IF jfnflgs.lhjb9  THEN 1B6 ELSE 0) %
                ;Protection %
                + (IF jfnflgs.lhjb10 THEN 1B5 ELSE 0) %
                ;Account %
                + (IF jfnflgs.lhjb11 THEN 4B4 ELSE 0) % ;T %
                + 1;              % with proper file punctuation
                %
            jfnflink( fjfn, $jfnname, R3);
        % get PC jfn and find out if this user can access
        this PC %
            IF NOT getpcjfn( fjfn, $pcjfn, $errstring, FALSE)
            THEN
                BEGIN   % this user can't access this file %
                *tstring* _ *jfnname*, " not deleted because
                [", *errstring*, "];
                IF flags THEN
                    BEGIN
                    *astr* _ *astr*, "   ***   ", *tstring*, CR,
                    LF;
                    GOTO getnxtjfn;
                    END
                ELSE
                    BEGIN
```

```
                    IF NOT SKIP !rljfn( fjfn) THEN NULL;
                    err( $tstring );
                    END;
                END;
            % now delete the file (and its partial copy) %
              IF NOT delflandpc( fjfn, pcjfn, $errstring) THEN
                BEGIN   % can't delete file (or partial copy) %
                *tstring* _ *jfnname*, " not deleted because ",
                *errstring*;
                IF flags THEN
                    BEGIN
                    *astr* _ *astr*, "   ***   ", *tstring*, CR,
                    LF;
                    GOTO getnxtjfn;
                    END
                ELSE
                    BEGIN
                    IF NOT SKIP !rljfn( fjfn) THEN NULL;
                    IF NOT SKIP !rljfn( pcjfn) THEN NULL;
                    err( $tstring );
                    END;
                END;
            % now tell the user we have deleted this file %
              *astr* _ *astr*, "   ", *jfnname*;
              IF pcjfn THEN *astr* _ *astr*, " and its partial
              copy";
              *astr* _ *astr*, CR, LF;
            % now get rid of the jfn for the partial %
              IF NOT SKIP !rljfn( pcjfn ) THEN NULL;
            % now get the next file in the group %
              (getnxtjfn):                                    5I2B8A8A
              R1.LH _ jfnflgs;
              R1.RH _ fjfn;
              IF NOT SKIP !gnjfn( R1 ) THEN EXIT LOOP;
            END;
        % now get rid of any lingering jfns %
            IF NOT SKIP !rljfn( fjfn ) THEN NULL;
            IF NOT SKIP !rljfn( pcjfn ) THEN NULL;
        % all done now, so return %
           RETURN;
        END.   %%
```

```
(cdelgro)                %Core NLS Delete Group Command%      5I2C
   PROCEDURE (stid1, stid2, vsptr);
   LOCAL newstid;
   REF vsptr;

   stid1 _ grptst(stid1, stid2 :stid2);
   IF (newstid _ getnxt(getend(stid2))) = endfil THEN newstid
   _ gethck(stid1);
   IF NOT &vsptr THEN
      BEGIN
      IF NOT remgrp(stid1, stid2) THEN
         ABORT(baddelete,$"Illegal delete");
      delgrp(stid1, stid2, newstid);
      END
   ELSE mvdlfgrp( stid1, stid2, &vsptr, newstid, deltflag,
   FALSE, FALSE);
   RETURN;
   END.

(cdelmar)                %Core NLS Delete Marker Command%     5I2D
   PROCEDURE (tp1, tp2, fileno);
   LOCAL mname, count, char;
   LOCAL STRING astrng[25];
   REF tp1, tp2;
   *astrng* _ tp1 tp2;
   astrng.L _ MIN(5, astrng.L);
   CCPOS SF(*astrng*);
   count _ mname _ 0;
   LOOP
      BEGIN
      CASE char _ READC OF
         = ENDCHR: EXIT LOOP;
         ENDCASE BUMP count;
      CASE count OF
         = 1: mname.chr0 _ char;
         = 2: mname.chr1 _ char;
         = 3: mname.chr2 _ char;
         = 4: mname.chr3 _ char;
         = 5: mname.chr4 _ char;
         ENDCASE ABORT(longmarker,$"Marker name too long");
      END;
   delmkn (fileno, mname);
   RETURN;
   END.

(cdelmodfil)   %Core NLS Delete Modifications to File Command%
                                                          5I2E
   PROCEDURE (fileno);
   unlkfile (fileno, TRUE);
   RETURN;
   END.

(cdelsta)                %Core NLS Delete Statement Command% 5I2F
   PROCEDURE (stid, vsptr);
   REF vsptr;
```

```
        IF getsub(stid) # stid THEN ABORT(baddelete,$"Illegal
        delete");
        cdelgro(stid, stid, &vsptr);
        RETURN;
        END.

     (cdeltex)              %Core NLS Delete Text Command%      5I2G
        PROCEDURE (bug1, bug2, includespace);
        REF bug1, bug2;
        IF includespace THEN incspc(&bug1, &bug2);
          %msntx();%
        cldtxt(&bug1, &bug2);
        ST bug1 _ SF(bug1) bug1, bug2 SE(bug1);
          %msftx();%
        RETURN;
        END.

     (incspc) PROCEDURE (ptr1,ptr2);                            5I2H
        REF ptr1, ptr2;
        FIND ptr2 > (SP ^ptr2 / ptr1 < (SP ^ptr1 / TRUE));
        RETURN;
        END.

%deliver an NSW file%
% entry   - called by WM to pass startup info%
   (xentry) PROCEDURE  %Called by NSW-WM at Runtool time.  Saves
   project-node string for semaphore comparison and generation of PC
   and initials file names.  DO NOT CALL A WM ROUTINE IN THIS
   PROCEDURE.%                                                  5K1
      ( wmusrno,  %User-id (INTEGER) %
      wmprojnode REF,  %addr of project+node name (CHARSTR)%
      wmtoolno  %tool-use name (INTEGER) %
      );

      %+NSW%                                                    5K1F
      nswuser _ wmusrno;
      *nswpplusn* _ *wmprojnode*;
      FIND  SF(*nswpplusn*) ^p1 [*+1 ^p3 ^p2 _p2 SE(*nswpplusn*)
      ^p4;
      *nswproject* _ p1 p2;
      *nswnode* _ p3 p4;
      %+NSW%                                                    5K1L
      RETURN;
      END.
%expand window%
   (xexpand)  %Execute Expand window Command%                    5L1
      PROCEDURE (
         %FORMALS%
            sourceptr REF,   %source pointer%
            destptr REF      %destination pointer%
            );
      dpset(dspno, endfil, endfil, endfil);
      movbndry(&sourceptr, &destptr);
      RETURN;
      END.
```

```
%expunge%
   %expunge PCP interface routine%
      (xexpunge)   % CL: ;  Execute Expunge Command %
      PROCEDURE (entity REF);                                          5M1A
         % Procedure description
            FUNCTION
               This is the protocol interface procedure for the
               expunge command.  Currently only expunge directory is
               supported.
            ARGUMENTS
               entity--REF-addr of LSEL list.  Command word
               identitfies type of expunge.  Currently only
               directories can be expunged.
            RESULTS
               proc-value always true.
            NON-STANDARD CONTROL
               none
            GLOBALS
               Reads cwtype
            %
         % Declarations %
         % process by type of expunge %
            CASE ELEM #entity#[cwtype] OF
               = 9 %- directory -%:
                  cexpdir(0);  %expunge connected directory%
               ENDCASE err(notyet);
         % Return %
            RETURN;
         END.

   %expunge core routines%
      (cexpdir)   % GB: core NLS Expunge Directory procedure %
      PROCEDURE (exptype % => no value %);                             5M2A
         % Procedure description
            FUNCTION
               Expunges (i.e. really throws away files) from both
               the connected and the login directories.  Works by
               first expunging the connected directory, then
               connecting to the login directory, then expunging
               that, then reconnecting to the previously connected
               directory.  The login directory is also expunged
               because NLS puts Partial Copies of all files on the
               login directory.
               This has a potential pitfall: if connecting to a
               directory requires a password, this procedure will
               cause an error because it doesn't know the password.
               It should only be used on directories that do not
               require a password when connecting to them.
            ARGUMENTS
               exptype - INTEGER - type of expunge:
                  0   - all deleted files (for any reason)
                  487 - nonexistent files
                  287 - explicitly deleted files
                  486 - scratch files
                  286 - temporary files
            RESULTS
```

```
        none
    NON-STANDARD CONTROL
        Error if can't connect to login directory or
        reconnect to connected directory
    GLOBALS
        chbmty - read only
    EXAMPLE
        cexpdir(486)
    %
% Declarations %
    LOCAL
        logdirnum,      % login directory number %
        condirnum,      % connected directory number %
        bytptr;         % byte pointer for GTPSW and CNDIR
        jsies %
% get login and connected directory numbers %
    !gjinf();
    logdirnum _ R1.RH;
    condirnum _ R2.RH;
% connect to login directory (and expunge it) if needed %
    IF logdirnum # condirnum THEN
        BEGIN
        % connect to login directory %
            IF NOT SKIP !cndir( logdirnum, 0) THEN
                err( $"can't connect to login directory" );
        % now expunge the login directory %
            cexpcondir (exptype);
        % now reconnect to the original connected directory %
            bytptr _ chbmty + $jpassw;
            IF NOT SKIP !cndir( condirnum, bytptr) THEN
                err( $"can't reconnect to connected directory"
                );
        END;
% expunge the connected directory %
    cexpcondir (exptype);
% we're done, so return %
    RETURN;
END.   %%
```

```
(cexpcondir)   % GB: core NLS Expunge Connected Directory
procedure %
PROCEDURE (exptype % => no value %);                              5M2B
   % Procedure description
      FUNCTION
         Like 'cexpdir', except expunges only the connected
         directory, not the login directory.
      ARGUMENTS
         exptype - INTEGER - type of expunge:
            0    - all deleted files (for any reason)
            4B7 - nonexistent files
            2B7 - explicitly deleted files
            4B6 - scratch files
            2B6 - temporary files
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      EXAMPLE
         cexpcondir(4B6)
      %
   % expunge the connected directory %
      !gjinf();  %directory number in R2%
      R2.LH _ exptype;
      !deldf(R2);
   % we're done, so return %
      RETURN;
   END.   %%
```

```
%force%
   %force PCP interface routine%
      (xforce) %Force case Set Command%                                5N1A
         PROCEDURE (
            %FORMALS%
               entity REF,   %parameter one%
               caseptr REF,  %case to force to%
               destptr REF   %destination pointer%
               );
            LOCAL retry, destination REF, csize, hinc, vinc, da REF,
            end1, save, tp2 REF, stid, adstr[40], case;
            LOCAL STRING sizestring[10], badmsg[200];
         %----------------------%
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         IF NOT nodisplay THEN dpset(dspno, endfil, endfil, endfil);
         case _ CASE ELEM #caseptr#[cwtype] OF
            = 51 %- upper -%: upcase;
            = 52 %- lower -%: lowcase;
            = 53 %- first letter -%: iupcase;
            = 54 %- sentence -%: supcase;
            ENDCASE xsmode;
         CASE ELEM #entity#[cwtype] OF
            = 30 %- link -%:
               BEGIN
               lnkprs( &destination, $adstr);
               destination _ adstr[1s];
                  destination[1] _ adstr[1s+1];
               [&destination+d2sel] _ adstr[1e];
                  [&destination+d2sel+1] _ adstr[1e+1];
               END;
            ENDCASE;
         CASE ELEM #entity#[cwtype] OF
            = 2 %- character -%, = 3 %- word -%, = 4 %- visible -%,
            = 11 %- invisible -%, = 30 %- link -%, = 8 %- integer
            -%, = 1 %- text -%:
               BEGIN
               IF NOT nortnrings THEN
                  clist(ctcmk, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dsprfmt, destination, endfil, destination);
               curmkr _ destination;  curmkr[1] _
               destination[d2sel+1]-1;
               csetctex(&destination, &destination+d2sel, case);
               IF NOT nortnrings THEN clupdt();
               END;
            = 29 %- statement -%:
               BEGIN
               IF NOT nortnrings THEN
                  clist(ctcfm, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dsprfmt, destination, endfil, destination);
               curmkr _ destination;  curmkr[1] _ 1;
               csetcsta(destination, case);
               IF NOT nortnrings THEN clupdt();
               END;
```

```
            = 27 %- group -%, = 28 %- plex -%, = 26 %- branch -%:
               BEGIN
               IF NOT nortnrings THEN
                  clist(ctcfm, destination.stfile, nofile);
               IF NOT nodisplay THEN
                  dpset(dspallf, destination, endfil, endfil);
               curmkr _ destination;  curmkr[1] _ 1;
               csetcgro(destination, [&destination+d2sel], case);
               IF NOT nortnrings THEN clupdt();
               END;
            = 51 %- mode -%:
               csetcmod(case);
            ENDCASE
            IF NOT donthelp THEN
               BEGIN
               *badmsg* _ "The type argument was incorrect.  The
               allowed values are Branch, Character, Group, ";
               *badmsg* _ *badmsg*, "Integer, Invisible, Link, Mode,
               Plex, Statement, Text, Visible, or Word.";
               retry _ HELP (badarg, $badmsg, cindex);
               REPEAT CASE (retry);
               END
            ELSE ABORT (badarg, $"The type argument was
            incorrect.");
         RETURN;
         END.

%force core routines%
   (csetcgro)              %Core NLS Set Case Group Command%   5N2A
      PROCEDURE(stid1, stid2, type);
        %msntx();%
      stid1 _ grptst(stid1, stid2 :stid2);
      stid2 _ getend(stid2);
      IF stid1 # stid2 THEN
         DO csetcsta(stid1, type) UNTIL (stid1 _ getnxt(stid1)) =
         stid2;
      csetcsta(stid2, type);
        %msftx();%
      RETURN;
      END.

   (csetcmod)              %Core NLS Set Case Mode Command%    5N2B
      PROCEDURE (type);
      CASE type OF
         = upcase, = lowcase, = iupcase, = supcase: xsmode _
         type;
         ENDCASE ABORT(badcase, $"Illegal case mode");
      RETURN;
      END.

   (csetcsta)              %Core NLS Set Case Statement Command%
                                                          5N2C
      PROCEDURE(stid, type);
      LOCAL TEXT POINTER ptr1, ptr2;
        %msntx();%
      FIND SF(stid) ^ptr1 SE(stid) ^ptr2;
```

```
      CASE type OF
         =upcase: % capital %
            ST stid _ +ptr1 ptr2;
         =lowcase: % lower %
            ST stid _ -ptr1 ptr2;
         =iupcase: % initial upper %
            iupper ($ptr1, $ptr2);
         =supcase: %sentence upper case %
            supper ($ptr1, $ptr2);
         ENDCASE  ABORT(badcase,$"Illegal Case Mode");
      %msftx();%
      RETURN;
      END.
   (iupper) % iupper edits the text string between text pointers
   ptr1 and ptr2, forcing the first alpha of each word to upper
   case %                                                    5N2D
      PROCEDURE( ptr1, ptr2 );
         LOCAL i, alphaflag, char;
         LOCAL STRING temp[2000];
         REF ptr1, ptr2;
         alphaflag _ FALSE;
         *temp* _ ptr1 ptr2; %make local copy of string%
         FOR i _ 1 UP UNTIL > temp.L DO
            CASE char _ *temp*[ i ] OF
               IN ['a, 'z]:
                  BEGIN
                  IF NOT alphaflag THEN *temp*[ i ] _ char - 40B;
                  alphaflag _ TRUE;
                  END;
               IN ['A, 'Z]:
                  BEGIN
                  IF alphaflag THEN *temp*[ i ] _ char + 40B;
                  alphaflag _ TRUE;
                  END;
               = '': %no caps after apostrophe inside word%
                  NULL;
            ENDCASE alphaflag _ FALSE;
         ST ptr1 ptr2 _ *temp*; % replace edited string %
         RETURN;
         END.
   (supper) % supper edits the text string between text pointers
   ptr1 and ptr2, forcing the first alpha of each sentence
   (preceded by a period, question mark, or exclamation point and
   one or more spaces) to upper case %                       5N2E
      PROCEDURE( ptr1, ptr2 );
         LOCAL i, periodflag, sentenceflag, char;
         LOCAL STRING temp[2000];
         REF ptr1, ptr2;
         sentenceflag _ periodflag _ TRUE;        %Assume prt1
         points to the start of a statement, and so the first
         alpha character should be forced to upper case%
         *temp* _ ptr1 ptr2; %make local copy of string%
         FOR i _ 1 UP UNTIL > temp.L DO
            CASE char _ *temp*[ i ] OF
               = '., = '?, = '!:
                  periodflag _ TRUE;
```

```
                    = SP:
                       BEGIN
                       IF periodflag THEN sentenceflag _ TRUE;
                       periodflag _ FALSE;
                       END;
                    ENDCASE
                       BEGIN
                       IF sentenceflag THEN *temp*[i] _ char - 40B;
                       periodflag _ sentenceflag _ FALSE;
                       END;
              ST ptr1 ptr2 _ *temp*; % replace edited string %
              RETURN;
              END.
        (csetctex)              %Core NLS Set Case Text Command%     5N2F
           PROCEDURE (bug1, bug2, type);
           REF bug1, bug2;
             %msntx();%
           CASE type OF
              =upcase: % capital %
                 ST bug1 _ SF(bug1) bug1, +bug1 bug2, bug2 SE(bug1);
              =lowcase: % lower %
                 ST bug1 _ SF(bug1) bug1, -bug1 bug2, bug2 SE(bug1);
              =iupcase: % initial upper %
                 iupper (&bug1, &bug2);
              =supcase: %sentence upper case%
                 supper (&bug1, &bug2);
              ENDCASE ABORT(badcase,$"Illegal Case Mode");
             %msftx();%
           RETURN;
           END.

%freeze%
   %freeze PCP interface routine%
      (xfreeze) %Execute Freeze Command%                              501A
         PROCEDURE (
            %FORMALS%
               destptr REF,  %destination pointer%
               vs REF        %viewspec pointer%
               );
            LOCAL da REF, destination REF, vswrd1, vswrd2;    % ptr
            to display area %
         %----------------------%
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         &da _ dsparea(ELEM #destptr#[wndw]);
         IF NOT &vs THEN
            BEGIN
            vswrd1 _ da.davspec;
            vswrd2 _ da.davspc2;
            END
         ELSE
            BEGIN
            vswrd1 _ vs;
            vswrd2 _ vs[1];
            END;
         IF NOT nortnrings THEN
```

```
            IF da.davspec.vsfrzf THEN
               dpset(dspstrc,destination,endtil,endfil)
         ELSE dpset(dspno,endfil,endfil,endfil);
         cfresta (&da, destination, vswrd1, vswrd2);
         RETURN;
         END.

   %treeze core routines%
      (cfresta)                                                    502A
         %Given the addrss of a display are, an stid, and two
         viewspec words, this routine will create a frozen element
         for the stid passed it, in the display area passed.  If
         the stid is already on the frozen list for this area, the
         new vspec words will replace the old ones for that
         element.%
         %------------------%
         PROCEDURE(dpa, stid, vspec1, vspec2);
         LOCAL frzelm;
         REF dpa, frzelm;
         IF &frzelm _ dpa.dafrzl THEN
            BEGIN
            LOOP
               BEGIN
               IF frzelm.fzstid = stid THEN
                  BEGIN
                  frzelm.fzvspec _ vspec1;
                  frzelm.fzvspc2 _ vspec2;
                  RETURN;
                  END;
               IF frzelm.fznext THEN &frzelm _ frzelm.fznext
               ELSE EXIT;
               END;
            IF NOT fzfree THEN ABORT(5, $"System error.  String
            variable overflowed.");
            &frzelm _ frzelm.fznext _ fzfree := [fzfree].fznext;
            frzelm.fznext _ 0;
            END
         ELSE
            BEGIN
            IF NOT fzfree THEN ABORT(5, $"System error.  String
            variable overflowed.");
            dpa.dafrzl _ &frzelm _ fzfree := [fzfree].fznext;
            frzelm.fznext _ 0;
            END;
         frzelm.fzstid _ stid;
         frzelm.fzvspec _ vspec1;
         frzelm.fzvspc2 _ vspec2;
         frzelm.fzexis _ TRUE;
         RETURN;
         END.

%Frontend termination rule%
   %FE terminate PCP interface routine%
      (xfterm) % FE Termination Procedure %                        5P1A
         PROCEDURE ;
         %---------------------%
```

```
        %No tool cleanup%
           csupdate _ 0;
        RETURN;
        END.

%gettext%
   %gettext PCP interface routine%
      (xgettext)   %Execute Getinput Command%                          501A
         PROCEDURE (
            %FORMALS%
               sourceptr REF,%pointer to source of input%
               count,        %number of characters to return%
               rtnlist REF); %return arg%
            LOCAL TEXT POINTER stmt1, stmt2;
            LOCAL retry REF, srcstid, endstid;
            LOCAL STRING newstmt[1000];

            %--------------------%
            &retry _ &sourceptr;  %To allow HELP return to work%
            %Get an STID for the source%
            CASE ELEM #retry#[cwtype] OF
               %   Branch inoperable, note double percents
               = 7 %%- oldfilename -%%:
                  BEGIN %%Load the old file %%
                  ....
                  srcstid _ XXXXX;      %%STID of origin%%
                  endstid _ endfil;
                  srcstid _ getnxt(srcstid;     %%STID of first
                  statement%%
                  END;
               %
               = 29 %- statement -%:
                  srcstid _ endstid _ [ELEM #retry#[tppair]];
               = 26 %- branch -%, = 27 %- group -%, = 28 %- plex -%:
                  BEGIN
                  srcstid _ [ELEM #retry#[tppair]];
                  endstid _ getend([ELEM #retry#[tppair] + 2]);
                  END;
               ENDCASE
               IF NOT donthelp THEN
                  BEGIN
                  &retry _ HELP (badarg, $"The selection type was
                  incorrect.  The allowed values are Branch, Group,
                  plex, or Statement.", cindex);
                  REPEAT CASE;
                  END
               ELSE ABORT (badarg, $"The selection type was
               incorrect.");
            FIND SF(srcstid) ^ stmt1 SE(srcstid) ^ stmt2;
            LOOP        %Temporarily return min(one statement, count
            characters) -- except if count = 0 return one statement%
               BEGIN
               *newstmt* _ stmt1 stmt2;
               IF (newstmt.L > count AND count > 0) THEN
                  BEGIN %Don't return the whole statement %
                  *newstmt* _ *newstmt*[1 TO count];
```

```
            stmt1[1] _ stmt1[1] + count;
            END
         ELSE
            BEGIN
            IF srcstid = endstid THEN EXIT LOOP;
            srcstid _ getnxt(srcstid);
            FIND SF(srcstid) ^ stmt1 SE(srcstid) ^ stmt2;
            END;
         count _ coreturn($newstmt);
         END;      %of loop %
      #rtnlist#[1] _ *newstmt*;
      RETURN;
      END.
%goto%
   %goto PCP interface routine%
      (xgoto) %Prepares the back end for GOTO Subsystem Command%  5R1A
         PROCEDURE (
            %FORMALS%
               enttype,  % TRUE if STRING, FALSE if LSEL %
               entity REF, % string or LSEL(#"OLDFILENAME")
               containing subsystem name%
               rtnlist REF LIST %to store return result: Grammar
               name%
               );
            LOCAL i, entype;
            LOCAL tp1 REF, tp2 REF, dirname REF;
            LOCAL TEXT POINTER txptr, tpfs, tpfe, tpcomma;
            LOCAL adstr[40];
            LOCAL STRING subname[70], tempstr[70], errstr[1999];
            LOCAL LIST loclist[1];
         %----------------------%
         IF enttype THEN
            BEGIN  % subsys name from command word string %
            *subname* _ *entity*;
            % Package must already exist, just open it %
               IF i _ pkgidx($subname : &dirname) THEN
                  BEGIN
                  opnpkg(i);
                  prvpkg _ curpkg := i;
                  % build grammar name for return %
                     *subname* _ '<, *dirname*, '>, *subname*;
                     #rtnlist#[1] _ USE (rtnstring($subname));
                  END;
            % return null to show no updated subslist %
               #rtnlist#[2] _ ;
            END
         ELSE
            BEGIN % LSEL(#"OLDFILNAME") %
            % parse filename %
               &tp1 _ ELEM #entity#[tppair];
               &tp2 _ &tp1 + d2sel;
            %  build full grammar name in TENEX format %
               lnbfls(&tp1, $adstr, $subname);
            % attempt to load the subsystem %
               % load the program into the user programs buffer%
                  IF cldprog($adstr, TRUE %display messages%,
```

```
                $errstr) THEN
                BEGIN % bad loading %
                   *errstr* _ "Cannot Load ", *subname*, "
                   SUBSYSTEM", CR, LF, *errstr*;
                   err($errstr);  %err does not return%
                END;
                IF errstr.L THEN typelit(1, $errstr);
             % get just the subsys name %
                tpfs _ adstr[fs];  tpfs[1] _ adstr[fs + 1];
                tpfe _ adstr[fe];  tpfe[1] _ adstr[fe + 1];
                *subname* _ + tpfs tpfe;
             % open the package %
                IF i _ pkgidx($subname : &dirname) THEN
                   BEGIN
                   opnpkg(i);
                   prvpkg _ curpkg := i;
                   % return dirname, subsys name to FE %
                      *subname* _ "<, *dirname*, ">, *subname*;
                      astruc($subname); %capitalize it%
                      #rtnlist#[1] _ USE (rtnstring($subname));
                   END
                ELSE err($"subsystem internal name does not
                match?");
          % build subsys list %
             listsubs($loclist);
             #rtnlist#[2] _ LIST(COPY #loclist#);
          END;
       #loclist# _ ;
       RETURN;
       END.
    (xsubsget)   % CL: ;  return list of available subsystems to
    FE %
    PROCEDURE (rtnlist REF);                                    5R1B
       % Procedure description
          FUNCTION
             This Backend xroutine determines the currently
             available (loaded) subsystems and returns them as a
             list of strings to the FE to be used as command words
             in the grammar.
          ARGUMENTS
             rtnlist--REF-pointer to result list, used for
             returning results to the FE.
          RESULTS
             proc-value--Always true.
          NON-STANDARD CONTROL
             Notes of type return and unwind will be caught while
             building the list; list elements will be freed.
          GLOBALS
             none
          %
       % Declarations %
          LOCAL LIST
             loclist[1];
       % Copy names from packages %
          INVOKE(catlnull);
          listsubs($loclist);
```

```
        % Return %
           #rtnlist#[1] _ LIST(COPY #loclist#);
           #loclist# _ ;
           DROP (catlnull);
           RETURN;
        % catchphrases %
           (catlnull)  CATCHPHRASE();                                    5R185A
              BEGIN
              CASE SIGNALTYPE OF
                 = notetype:
                    CASE SIGNAL OF
                       = return, = unwind :
                          BEGIN
                          DISABLE (catlnull);
                          NULL-LISTS;
                          END;
                    ENDCASE;
                 ENDCASE;
              CONTINUE;
              END;
        END.

     (listsubs)   % CL: ;  return list of available subsystems %
     PROCEDURE (slist REF);                                              5R1C
        % Procedure description
           FUNCTION
              Fetches the currently loaded subsystems from the
              Middle End package table and constructs a list of
              strings of the subsystem names.
           ARGUMENTS
              slist--REF-the addr of the resulting list of strings.
           RESULTS
              proc-value--always TRUE.
           NON-STANDARD CONTROL
              none
           GLOBALS
              Reads $pkgs, npkgs
           %
        % Declarations %
           LOCAL
              pkg REF,
              ielem _ 1, ipkg;
        % Copy names from packages %
           &pkg _ $pkgs;  % get addr of package table %
           FOR ipkg _ 1 UP UNTIL > npkgs DO
              BEGIN
              IF pkg.pkexis THEN
                 BEGIN  % append subsys string %
                 #slist# !_
                    USE makedesc(ustring, pkg.pkname, FALSE);
                 END;
              &pkg _ &pkg + pkgdesc.SIZE;
              END;
        % Return %
           RETURN;
        END.
```

```
%handle%
   %handle PCP interface routine%
      (xhandle) PROCEDURE;                                          5S1A
         RETURN;
         END.
%initialization from FE%
   (xinit)  % CL:GB;  Initialization procedure called in FE initrule
   %
   PROCEDURE (idstr REF, rtnlist REF % => see arg %);               5T1
      % Procedure description
         FUNCTION
            This procedure is called during the initialization rule
            of the BASE grammar.
            If 'dspcmd' has been allocated, this is a Goto Base or
            Exec Base command and return without any processing.
            The FE will restart a failing init rule, so this
            procedure always returns successfully to the FE and
            returns the real outcome as a result.  The result is
            tested in the grammar and if it indicates failure, a
            parsefunction to terminate the tool is called.
            Allocate lists for display package command list and
            argument list for FE calls.  Must be done here so that
            'dismes' can be called during initialization to display
            error messages to user if necessary.
            Get user ident, initialize display areas and load
            initials file.
            Check for new journal mail.  If new journal mail found
            turn off new journal mail bit in initial file and return
            message to FE for display to user.
            Check for entry subsystem.  If there is one, return name
            of subsystem to FE.
            Check for startup commands branch.  If there is one,
            return name of sequential file holding commands to FE.
         ARGUMENTS
            rtnlist -- REF - addr of result list (for middle end).
         RESULTS
            proc-value -- TRUE always
            rtnlist -- REF - addr of result list;
               LIST ( outcome BOOLEAN,
                  NULL / string --new journal mail
                  NULL / string --entry subsystem name
                  NULL / string --startup commad branch seq. file
                  name(TENEX))
         NON-STANDARD CONTROL
            SIGNALS CAUGHT
               Failure on startup commands branch address results in
               NULL element returned to FE.
               The catchphrase initfail returns a failure result to
               the FE, which calls a parsefunction to terminate the
               tool.
               All other signals are continued.
         GLOBALS
            uses 'tda' (assumes it to be address of display area for
            initial file)
            uses entsubsystem, stuplist
```

```
        SET
           dspcmd -- address of list for display package
           commands to FE batch-commands procedure
           dsparg -- address of list for display pacakge
           arguments to FE procedures
     %
  % Declarations %
     (endstid) LOCAL;                                          5T1B1
     (f1) LOCAL REF;                                           5T1B2
     (stupstr) LOCAL REF;                                      5T1B3
     (tp1) TEXT POINTER;                                       5T1B4
     (tp2) TEXT POINTER;                                       5T1B5
     (tp3) TEXT POINTER;                                       5T1B6
     (workstring) STRING [100];                                5T1B7
     REF tda, dspcmd, dsparg;
  % return immediately if not first time through %
     IF &dspcmd THEN
        BEGIN
        #rtnlist# _  % set up return list %
           USE makedesc(uboolean,TRUE, FALSE), NULL, NULL, NULL;
        RETURN;
        END;
  % invoke catchphrase %
     INVOKE(initfail, rtnsetfail);
  % Allocate display package lists %
     &dspcmd _ getlst(cmdlsize);
     &dsparg _ getlst(arglsize);
  % set user id, extra defensive for old FEs %
     IF &idstr AND idstr.L IN [1,20] AND idstr.M IN [1,20] THEN
        BEGIN % copy initials and put into cinit, packed into
        5-bit%
        *initsr* _ *idstr*;
        cinit _ setcinit($initsr);
        END;
  % Get information from works manager about user, e.g. user
  IDENT. %
     wminfo();
  % initialize user options  %
     uoinit();
  % initialize display %
     setdis();
  % load included subs/progs now that we have a display area
  (for CAs, SGs) %
     uoproginit();
  %get mode, system name%
     IF nlmode = typewriter THEN
        BEGIN
        *nlsnam* _ "TNLS";
        nlstyp _ tntyp;
        END
     ELSE
        BEGIN
        *nlsnam* _ "DNLS";
        nlstyp _ dntyp;
        END;
  %startup message -- intmsg string initialized in this file%
```

```
      IF intmsf THEN
         BEGIN
         % send INTMSG to FE for portrayal--  could be done at
         info exchange time %
         END;
   % open initial file %
      initid();
   % allocate block for correspondence list %
      clistaddr _ getblk ((clsize _ 300), $dspblk);  %100
      entries%
   % set up return list %
      %Successfull return result%
         #rtnlist# _
            USE makedesc(uboolean,TRUE, FALSE);
      %Journal mail result%
         &fl _ flntadr(tda.dacsp.stfile);
            %get file table entry address for initial file%
         !gtfdb (fl.florig, 1000024B, $R3); %check new mail bit%
         IF R3.ojdelf THEN %new mail%
            BEGIN
            R2 _ 0; R2.ojdelf _ 1;
            chnfdb (fl.florig, 24B, R2, 0);
            *workstring* _ "You have new Journal mail";
            #rtnlist# !_
               *workstring*  %return new journal mail msg%
            END
         ELSE
            #rtnlist# !_
               NULL;  %no new mail%
      %Entry subsystem result%
         IF entsubsystem .L THEN
            BEGIN  %there is an entry subsystem%
            IF FIND SE(*entsubsystem*) [":"] > CH ^tp1 THEN
               %subsystem name includes directory, omit it from
               string sent to FE%
                  *workstring* _ tp1 SE(*entsubsystem*)
            ELSE *workstring* _ *entsubsystem*;
            #rtnlist# !_
               *workstring*;  %entry subsystem name%
            END
         ELSE  %no entry subsystem%
            #rtnlist# !_
               NULL;
      %Startup commands branch file result%
         IF stuplist.L AND (&stupstr _ #stuplist#[1]) AND
         stupstr.L THEN
            BEGIN  %there is a startup commands branch%
            FIND SF(*stupstr*) ^tp1 SE(*stupstr*) ^tp2;
            tp3 _ curmkr;
            tp3[1] _ 1;
            INVOKE(badstup, stupfail);
            caddexp($tp1, $tp2, &tda, $tp3);
            DROP(badstup);
            endstid _ getend(tp3);
            cprocess(tp3, endstid, $workstring);
            #rtnlist#[4] _
```

```
            *workstring*;  %startup commands file name%
         END
      ELSE  %no startup commands branch%
         #rtnlist#[4] _
            NULL;
   % drop catchphrase %
    (stupfail):                                                    5T1P1
    DROP(initfail);
   % return %
    (rtnsetfail):                                                  5T1Q1
      RETURN;
   % catchphrases %
    (badstup) CATCHPHRASE();                                       5T1R1
      BEGIN
      CASE SIGNALTYPE OF
         = aborttype :
            BEGIN
            #rtnlist#[4] _ NULL;
            TERMINATE;
            END;
         ENDCASE;
      CONTINUE;
      END;
    (initfail) CATCHPHRASE;                                        5T1R2
      BEGIN
      CASE SIGNALTYPE OF
         =aborttype:
            BEGIN
            DISABLE (initfail);
            #rtnlist#[1] _ USE makedesc(uboolean, FALSE,
            FALSE);
            TERMINATE;
            END;
         ENDCASE
            CONTINUE;
      END;
   END.

%+NSW%                                                             5T2
(xnswinit)   % CL: ;  BASE initialization procedure %
PROCEDURE (project REF, node REF, rtnlst REF);                     5T3
   % Procedure description
      FUNCTION
         Called from the INITIALIZATION rule in the BASE grammar
         for NLS encapsulated under the FOREMAN in the NSW.  This
         procedure is necessary because the project and node are
         sent in the NSW.  It stores the project and node (forced
         to upper case ) and calls 'xinit'.
      ARGUMENTS
         project:  STRING variable --  logon project
         node:  STRING variable -- logon node
         rtnlst:  LIST variable -- return values
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
```

```
        GLOBALS
           none
        %
      % Declarations %
      % Save project and node and do initialization %
        nswproject _ getstring(project.L, $dspblk);
        *[nswproject]* _ *project*;
        nswnode _ getstring(node.L, $dspblk);
        *[nswnode]* _ *node*;
        astruc(nswproject);  %force to upper case%
        astruc(nswnode);  %force to upper case%
        xinit(&rtnlst);
      % Return %
        RETURN;
      END.

   %+NSW%                                                          5T4
%insert%
   %insert PCP interface routines%
      (xinsert)    %Execute Insert Command%  PROCEDURE (          5U1A
         %FORMALS%
            entity REF,                %entity type%
            destptr REF,               %destination pointer%
            level,             %level adjustment characters%
            sourceptr REF,     %source pointer%
            txtrtn,  %if TRUE, return text of statement%
            tprtn              %if TRUE, return text pointer(s)%
            );
         LOCAL
            destination REF, source REF, lnktp1 REF, lnktp2 REF,
            temp, endstid, grplist REF, type, da REF, cords, i, x,
            y, adstr[40], edgeloc, srcspace[20], srcdecoded REF,
            retry, enttype;
         LOCAL STRING
            linkstr[500],     % string for links %
            badmsg[200],
            locstr[500];      % string for date and time %
         LOCAL TEXT POINTER
            tp1, tp2;
         LOCAL LIST tplist [4], txtlist[1];
         %----------------------%
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         IF &sourceptr THEN &source _ ELEM #sourceptr#[tppair];
         CASE enttype _ ELEM #entity#[cwtype] OF
            = 30 %- link -%:
               BEGIN
               &lnktp1 _ &source+d1sel;
               &lnktp2 _ &source+d2sel;
               IF lnktp1.stastr THEN
                  BEGIN
                  *linkstr* _ lnktp1 lnktp2;
                  lnktp1.RH _ lnktp2.RH _ $linkstr;
                  IF NOT FIND SF(linkstr) $(SP/TAB) ("(/"</"--")
                  THEN
                     *linkstr* _ "<, *linkstr*;
                  IF NOT FIND SE(linkstr) < $(SP/TAB) (")/">) THEN
```

```
                    *linkstr* _ *linkstr*, ">;
                  END;
               lnkprs (&lnktp1, $adstr);
               lnktp1[0] _ adstr[ls];
                  lnktp1[1] _ adstr[ls+1];
               lnktp2[0] _ adstr[le];
                  lnktp2[1] _ adstr[le+1];
               END;
            ENDCASE;
         CASE enttype OF
            %text/structure entities%
               = 2 %- character -%, = 1 %- text -%, = 11 %-
               invisible -%:
                  BEGIN
                  IF NOT nortnrings THEN
                     clist(ctcmk, destination.stfile, nofile);
                  IF NOT nodisplay THEN
                     dpset(dsprfmt, destination, endfil,
                     destination);
                  curmkr _ destination[d2sel];
                     curmkr[1] _ destination[d2sel+1] +
                     source[d2sel+1]-source[1] - 1;
                  cinstex(&destination+d2sel, &source,
                  &source+d2sel, FALSE);
                  IF NOT nortnrings THEN clupdt();
                  IF txtrtn THEN #txtlist# _ curmkr;
                  END;
               = 3 %- word -%, = 4 %- visible -%, = 8 %- integer -%,
               = 30 %- link -%:
                  BEGIN
                  IF NOT nortnrings THEN
                     clist(ctcmk, destination.stfile, nofile);
                  IF NOT nodisplay THEN
                     dpset(dsprfmt, destination, endfil,
                     destination);
                  curmkr _ destination[d2sel];
                     curmkr[1] _ destination[d2sel+1] +
                     source[d2sel+1]-source[1];
                  cinstex(&destination+d2sel, &source,
                  &source+d2sel, TRUE);
                  IF NOT nortnrings THEN clupdt();
                  IF txtrtn THEN #txtlist# _ curmkr;
                  END;
               = 29 %- statement -%:
                  BEGIN
                  temp _ 0;
                  curmkr _ xcmst( &destination, level, &source,
                  copyflag, temp);
                  curmkr[1] _ 1;
                  IF txtrtn THEN #txtlist# _ curmkr;
                  destination _ curmkr;  %For returning handles to
                  ISI%
                  destination[1] _ 1;
                  END;
               = 26 %- branch -%, = 28 %- plex -%, = 27 %- group -%:
                  BEGIN
```

```
                temp _ 0;
                curmkr _ xcmgrp( &destination, level, &source,
                copyflag, temp);
                curmkr[1] _ 1;
                destination[1] _ 1;
                CASE level OF
                   = 0: destination _ getsuc(destination);
                   = -1: destination _ getsuc(destination);
                ENDCASE
                   BEGIN
                   FOR i _ 1 UP UNTIL > level DO
                      destination _ getup(destination);
                   destination _ getsuc(destination);
                   END;
                IF txtrtn THEN
                   BEGIN
                   &grplist _ alloclist(20);
                   endstid _ getend(curmkr);
                   #grplist# _ destination;
                   DO
                      BEGIN
                      #grplist# !_ getnxt(destination);
                      IF grplist.L = grplist.M THEN
                         &grplist _ alloclist(grplist.M + 20);
                      END
                   UNTIL ELEM #grplist#[grplist.L] = endstid;
                   END;
                END;
             = 59 %- date -%, = 58 %- time -%:
                BEGIN
                % date (and time) to string; set up pointers %
                   *locstr* _ NULL;
                   getdat( $locstr );
                   CASE enttype OF
                      =59 %- date -%:
                         BEGIN
                         IF NOT
                            (FIND SF(*locstr*) $PT (SP ^tp1))
                               THEN ABORT (baddate, $"Bad Date From
                               TENEX");
                         ST tp1 _ SF(tp1) tp1;
                         END;
                      ENDCASE;
                   FIND SF(*locstr*) ^tp1 SE(*locstr*) ^tp2;
                IF NOT nortnrings THEN
                   clist(ctcmk, destination.stfile, nofile);
                IF NOT nodisplay THEN
                   dpset(dsprfmt, destination, endfil, destination);
                curmkr _ destination[d2sel];
                   curmkr[1] _ destination[d2sel+1];
                cinstex(&destination+d2sel, $tp1, $tp2, TRUE);
                IF NOT nortnrings THEN clupdt();
                IF txtrtn THEN #txtlist# _ curmkr;
                END;
             = 53 %- sendmail -%: %form%
                BEGIN
```

```
            *locstr* _
               *sjtitle*, EOL, *sjcomment*, EOL,
               *sjauthor*, SP, *initsr*, EOL, *sjnumber*, EOL,
               *sjaction*, EOL, *sjinfo*, EOL, *sjsubcol*, EOL,
               *sjkeywords*, EOL, *sjhandling*, EOL,
               *sjrecording*, EOL, *sjhardcopy*, EOL,
               *sjrfc*, EOL, *sjobsoletes*, EOL,
               *sjaccess*, EOL, *sjupdates*, EOL,
               *sjlink*, EOL, *sjforward*, EOL,
               *sjmessage*, EOL, *sjbranch*, EOL,
               *sjplex*, EOL, *sjgroup*, EOL,
               *sjfile*, EOL, *sjsendit* ;
            FIND SF(*locstr*) ^tp1 SE(*locstr*) ^tp2;
            curmkr _ cinssta(destination, level, $tp1, $tp2);
            curmkr[1] _ 1;
            IF txtrtn THEN #txtlist# _ curmkr;
            destination _ curmkr;  %For returning handles to ISI%
            destination[1] _ 1;
            IF NOT nodisplay THEN dpset(dspstrc, curmkr, endfil,
            curmkr);
            END;
         ENDCASE
         IF NOT donthelp THEN
            BEGIN
            *badmsg* _ "The type argument was incorrect.  The
            allowed values are Branch, Character, Date, Edge, ";
            *badmsg* _ *badmsg*, "Group, Integer, Invisible,
            Link, Plex, Sendmail, Statement, Text, Time, Visible,
            or Word.";
            retry _ HELP (badarg, $badmsg, cindex);
            REPEAT CASE (retry);
            END
         ELSE APOPT (badarg, $"The type argument was
         incorrect.");
      %
      IF tprtn THEN
         BEGIN
         #tplist# _ destination, destination[1];
         IF NOT ((destination = curmkr) AND (destination[1] =
         curmkr[1])) THEN
            #tplist# !_ curmkr, curmkr[1];
         END;
      IF tprtn THEN
         IF txtrtn THEN RETURN(grplist, tplist)
         ELSE RETURN (empty, tplist)
      ELSE
         IF txtrtn THEN RETURN(grplist, empty)
         ELSE RETURN;
      %
      RETURN;
      END.


(xinsstatement) %Execute repeat Insert Statement%          5U1C
   PROCEDURE (
      %FORMALS%
```

```
            level,                  %level adjustment value%
            sourceptr REF,          %source text for stmt%
            txtrtn,         %if TRUE, return text of statement%
            tprtn           %if TRUE, return text pointer(s)%
            );
        LOCAL source REF;
        LOCAL LIST tplist[2], txtlist[1];
        %---------------%
        IF &sourceptr THEN &source _ ELEM #sourceptr#[tppair];
        curmkr _ cinssta(ecurmkr, level, &source, &source+d2sel);
        curmkr[1] _ 1;
        IF NOT nodisplay THEN dpset(dspstrc, curmkr, endfil,
        getnxt(curmkr));
        %
        IF tprtn THEN #tplist# _ curmkr, curmkr[1];
        IF txtrtn THEN #txtlist# _ curmkr;
        IF tprtn THEN
           BEGIN
           #tplist# _ source, source[1];
           IF NOT ((source = curmkr) AND (source[1] = curmkr[1]))
           THEN
              #tplist# !_ curmkr, curmkr[1];
           END;
        IF tprtn THEN
           IF txtrtn THEN RETURN(txtlist, tplist)
           ELSE RETURN (empty, tplist)
        ELSE
           IF txtrtn THEN RETURN(txtlist, empty)
           ELSE RETURN;
        %
        RETURN;
        END.

%insert core routines%
   (cinssta)                %Core NLS Insert Statement Command% 5U2A
      PROCEDURE (stid, rlevcnt, tp1, tp2);
      LOCAL newstid; %new stid%
      REF tp1, tp2;
      IF tp1 # tp2 THEN ABORT(badstmt,$"Illegal Statement");
        %msnst();%
      newstid _ newrng(stid.stfile);
      insgrp(stid, rlevcnt, newstid, newstid);
      ST newstid _ tp1 tp2;
        %msfst();%
      RETURN(newstid);
      END.

   (cisstr) % insert string as statement routine%
   PROCEDURE (stid, astring REF, rlevcnt);                    5U2B
      % accepts stid, address of string to be inserted as
      statement, and (unlike the old cis routine) a relative
      level count: successor, levsuc = 0, positive integer =
      number of levels up, levup = 1, levdown = -1. %
      LOCAL TEXT POINTER tp1, tp2; %new stid%
      FIND SF(*astring*) ^tp1 SE(*astring*) ^tp2;
      RETURN(cinssta(stid, rlevcnt, $tp1, $tp2));
```

```
        END.

    (cinstex)                %Core NLS Insert Text Command%      5U2C
        PROCEDURE(tp1, tp2, tp3, insertspace);
        REF tp1, tp2, tp3;
          %msntx();%
        IF insertspace THEN
            ST tp1 _ SF(tp1) tp1, SP,
               $tp2 tp3,     % $ => don't move markers %
               tp1 SE(tp1)
        ELSE
            ST tp1 _ SF(tp1) tp1,
               $tp2 tp3,     % $ => don't move markers %
               tp1 SE(tp1);
          %msftx();%
        RETURN;
        END.

%jump%
    %jump PCP interface routines%
        (xjump)          %Execute Jump Command%
        PROCEDURE( entity REF, destptr REF, vs REF, destwndw );    5V1A
            % Procedure description
                FUNCTION
                    Execute jump command
                ARGUMENTS
                    entity -- command word -- type of jump
                    destptr -- selection -- reference stid for jump
                    vs -- new viewspecs (or NULL)
                    destwndw -- window id for jump destination
                LOCALS
                    destination -- pointer to CML record for destination,
                    da -- display area that corresponds to srcwndw
                    start -- reference point for jumping
                    ptr1 ptr2 -- pointers to address expression
                    loctype -- type of jump
                    linktype -- TRUE if jump to link
                    locvs -- local viewspecs record
                    newvs -- local saving of viewspecs
                    adstr -- parsed link data structure
                    structype - TRUE if jump to up/ successor etc
                    param - parameter to be passed to getpr
                    cacode, usgcode - local storage for ca and sg
                    retry -- HELP rule result
                    t1 t2 -- general text pointers
                    csp -- text pointer to the current top statement in
                    da
                    srchtp1 srchtp2 -- text pointers to recent pattern
                    search.
                      *** Keep these in order so that they look like a
                      tppair.
                RESULTS
                    none
                NON-STANDARD CONTROL
                    all signals handled but CONTINUEd
                GLOBALS
```

```
        none
      %
    % Declarations %
      LOCAL destination REF, da REF, dstda REF, start REF,
      ptr1 REF, ptr2 REF, loctype, linktype, locvs[2],
      newvs[2], adstr[40], strctype, param, cacode, usqcode,
      srcwndw, retry;
      LOCAL TEXT POINTER t1, t2, csp,
        srchtp1, srchtp2;  %keep these in order%
      LOCAL STRING locstr[200], badmsg[250];
    % protect against all errors %
      INVOKE (catjump);
    % undo destptr into destination %
      IF &destptr THEN &destination _ ELEM #destptr#[tppair]
      ELSE
        BEGIN  %use previous search pattern%
        FIND SF(*conreg*) ^srchtp1 SE(*conreg*) ^srchtp2;
        &destination _ $srchtp1;
        END;
    % initialize locals %
      &dstda _ dsparea(destwndw);
      % srcwndw _ IF &destptr THEN #destptr#[wndw] ELSE
      destwndw; %
      &da _ &dstda; %should be: dsparea(srcwndw); %
      IF &vs THEN % viewspecs from user %
        BEGIN
          locvs _ vs;
          locvs[1] _ vs[1];
        END
      ELSE % viewspecs from da %
        BEGIN
          locvs _ da.davspec;
          locvs[1] _ da.davspc2;
        END;
      loctype _ ELEM #entity#[cwtype];
      strctype _ TRUE;
    % arrange for structural jump: up successor etc %
      CASE loctype OF
        = cwstatement:   param _ 0;
        = cwsuccessor,
        = cwpredecessor,
        = cwup,
        = cwdown,
        = cwhead,
        = cwtail,
        = cwend,
        = cwback,
        = cwnext: param _ loctype;
        = cworigin:
          BEGIN
            destination.stpsid _ origin;
            param _ 0;
          END;
        ENDCASE strctype _ FALSE;
    % make the jump if structural type %
      IF strctype THEN
```

```
            BEGIN
               IF param THEN getpr(param, 1, &destination);
               cjump(&destination, $locvs, &dstda);
               DROP(catjump);
               RETURN;
            END;
         % if you are here it is a jump relative to current position
         %
         % initialize variables %
            csp _ da.dacsp;
            csp[1] _ da.daccnt;
            &start _ $csp;
            linktype _ (loctype = cwlink);
            &ptr1 _ &destination;
            &ptr2 _ &destination + d2sel;
         % create the address expression %
            CASE loctype OF
               = cwlink:
                  IF NOT destination.stastr THEN % bugged link %
                     BEGIN
                     lnkprs( &destination, $adstr );
                     &ptr1 _ $adstr[ls];
                     &ptr2 _ $adstr[le];
                     &start _ &destination;
                     END;
               = cwfile:
                  BEGIN
                  lnkprs(&destination, $adstr);
                  &ptr1 _ $adstr[ls];
                  &ptr2 _ $adstr[fe];
                  *locstr* _ ptr1 ptr2, ",)";
                  END;
               = cwname:          *locstr* _ ptr1 ptr2;
               = 63 %filenamed%: *locstr* _ "(, ptr1 ptr2, ",)";
               = cwfstname:       *locstr* _ ".o*", ptr1 ptr2;
               = cwnxtname:       *locstr* _ "*, ptr1 ptr2;
               = cwextname:       *locstr* _ "$, ptr1 ptr2;
               = 68 %frstcntnt%: *locstr* _ ".o", "", ptr1 ptr2, "",
               "=C";
               = cwnxtcontent:    *locstr* _ ".n", "", ptr1 ptr2, "",
               "=C";
               = cwfstword:       *locstr* _ ".o", "", ptr1 ptr2, "",
               "=W";
               = cwnxtword:       *locstr* _ ".n", "", ptr1 ptr2, "",
               "=W";
               ENDCASE
                  % IF NOT donthelp THEN
                     BEGIN
                     *badmsg* _ "The type argument was incorrect.
                     The allowed values are Back, Down, End,
                     Extname, File, Filenamed, Filereturn,
                     Firstcontent, ";
                     *badmsg* _ *badmsg*, "Firstname, Firstword,
                     Head, Link, Name, ";
                     *badmsg* _ *badmsg*,  "Next, Nextcontent,
                     Nextname, Nextword, Origin, Predecessor,
```

```
                    Return, Statement, Successor, Tail, or Up.";
                    retry _ HELP (badarg, $badmsg, cindex);
                    REPEAT CASE (retry);
                    END
                ELSE % ABORT (badarg, $"The type argument was
                incorrect.");
        % set the pointers for non-link types %
            IF NOT linktype THEN
                BEGIN
                    FIND SF(*locstr*) ^t1 SE(*locstr*) ^t2;
                    &ptr1 _ $t1;  &ptr2 _ $t2;
                END;
            % at this point ptr1 and ptr2 delimit an address
            expression.  start points to a starting (text) pointer
            (either bugged or from da or from last search). locvs
            contains the appropriate viewspecs %
        % evaluate the address expression %
            newvs _ caddexp(&ptr1, &ptr2, &da, &start : newvs[1],
            cacode, usqcode%, usesrr%);
            usesrr _ fakesrr;
        % institute the viewspecs for links %
            IF linktype THEN
                BEGIN
                locvs _ newvs;
                locvs[1] _ newvs[1];
                cspcacode _ cacode;
                cspusqcod _ usqcode;
                END;
        % do the jump stuff %
            cjump( &start, $locvs, &dstda);
        % drop the catchphrase %
            DROP( catjump );
        % and return %
            RETURN;
        %Catchphrases%                                          5V1A17
            (catjump) CATCHPHRASE();                           5V1A17A
                BEGIN
                CASE SIGNALTYPE OF
                    = aborttype :  %stop cmdfinish from reformatting
                    display%
                        cspupdate _ cspcacode _ cspusqcod _ 0;
                    ENDCASE;
                CONTINUE;
                END;
        END.

    (updcsp) PROCEDURE;                                        5V1B
        %Set cspupdate before calling xjump so non-link jumps to
        locations specified in links will use the viewspecs in the
        link in DNLS%
        cspupdate _ lda();
        RETURN;
        END.
    (xjumpreturn)   % LB: ; Execute  jump RETURN or jump FILE
    RETURN %
    PROCEDURE (entity REF, index, window);                     5V1C
```

```
% Procedure description
   FUNCTION
      Execute jump RETURN or jump FILE RETURN.  This
      procedure gets called after the user has seen the
      results of xgetjump ring and has indicated which
      statement or file to return to.  This procedure
      replaces the jump RETURN and jump FILE RETURN code in
      xjump.
   ARGUMENTS
      entity:  type of jump - values:
         62:  return
         67: file return
      index:  index into statement or file return ring
      window:  window id of last window bugged
   RESULTS
      none
   NON-STANDARD CONTROL
      none
   GLOBALS
      Sets cspvs, cspcacode, cspusqcod, cspupdate
   %
% Declarations %
   LOCAL srr, da REF, filestraddr;
   LOCAL TEXT POINTER tp1;
   LOCAL STRING locstr[200];
% this could be made to work across window boundaries if
use two window parameters, dstwindow as the destination and
use prevwindow as the source %
% Get currend da and set some globals from it %
   &da _ cspupdate _ dsparea(window);  %get current da%
   cspcacode _ da.dacacode;  %content analyzer program%
   cspusqcod _ da.dausqcod;  %sequence genarator program%
% Set up for jump RETURN or jump FILE RETURN %
   CASE ELEM #entity#[cwtype] OF
      = 67 %- filereturn -%:
         BEGIN
         %get the index-th entry in the file return ring%
            INVOKE (badfrr);
            filestraddr _ readfrring(da.dalink,index :
            usesrr);
            DROP (badfrr);
         %make the top of the statement return ring the
         current statement%
            curmkr _ readsrring(usesrr, 0 : curmkr[1],
            cspvs, cspvs[1]);
         %load requested file%
            FIND SF(*[filestraddr]*) ^tp1;
            CASE lnbfls( $tp1, 0, $locstr) OF
               = lhostn: NULL;
               ENDCASE
                  ABORT(noremote,$"Remote File
                  Manipulations Not Implemented Yet");
            curmkr.stfile _ cloafil($locstr);
         END;
      = 62 %- return -%:
         BEGIN
```

```
            %get the index-th entry in statement return ring
            and make it the current statement%
            INVOKE (badfrr);
            readfrring(da.dalink, 0 : srr);
            DROP (badfrr);
            curmkr _ readsrring(srr, index : curmkr[1], cspvs,
            cspvs[1]);
            END;
         ENDCASE ABORT (badarg, $"The type argument was
         incorrect.");
      % Set globals for display%
         IF NOT nodisplay THEN
         dpset(dspjpf, curmkr, endfil, endfil);
      % Return %
         RETURN;
      (badfrr) CATCHPHRASE();                                    5V1C8
         BEGIN
         DISABLE (badfrr);
         CASE SIGNALTYPE OF
            = notetype :  NULL;
            = helptype :  NULL;
            = aborttype :
               IF SIGNAL = frremptyentry THEN
                  err($"No current entry in file return ring.");
            ENDCASE;
         CONTINUE;
         END;
      END.

   (xjumpaddr) %Execute Jump to Address Relative Command%     5V1D
      PROCEDURE (
         %FORMAL ARGUMENTS%
            entity REF,    %type of jump%
            destptr REF,   %statement relative to%
            addrptr REF,   %relative address text%
            vs REF,        %viewspecs%
            window         %window number%
            );
         LOCAL destination REF, reladdr REF;
         %------------------%
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         IF &addrptr THEN &reladdr _ ELEM #addrptr# [tppair];
         %Get the resultant (relative) address -- the text
         pointer poiinted to by destination is actually changed
         to point to the relative address%
         caddexp (&reladdr, &reladdr+d2sel, dsparea(window),
         &destination);
         %Call xjump to do the work%
         xjump (&entity, &destptr, &vs, window);
         RETURN;
         END.

   (xgetjumpring)   % LE: ;  get jump ring entries %
   PROCEDURE (entity REF, window, rtnlst REF );                  5V1E
      % Procedure description
         FUNCTION
```

```
        Provides feedback for stepping through the rings for
        jump RETURN and jump FILE RETURN commands.  A FE
        parse function interacts with the user displaying the
        strings returned by this procedure.
     ARGUMENTS
        entity:  entity type
           values 62 (return) or 67 (file return)
        rtnlst:  address of list in which to store strings to
        be displayed to user
           for jump RETURN the strings are beginning text of
           the statement in the statement return ring
           for jump FILE RETURN the strings are the file
           names of the files in the file return ring
        window:  window id of last window bugged
     RESULTS
        fills rtnlst
     NON-STANDARD CONTROL
        Go to err if
           no statments/files in return ring
           no text block for associated with node
     GLOBALS
        none
     %
  % Declarations %
     LOCAL da REF, stid, stdb, len, cc, pvs1, pvs2, srr REF,
     filestraddr, retry, count;
     LOCAL TEXT POINTER tp1,tp2;
     LOCAL STRING temp[200];
  % Set da to current display area %
     &da _ dsparea(window);
  % Get statement texts or file names according to requested
  entity %
     CASE ELEM #entity#[cwtype] OF
        = 62: %- return -%
           BEGIN
           INVOKE (badfrr);
           readfrring(da.dalink, 0 : &srr);  %get st. ring%
           DROP (badfrr);
           IF ( count _ srrlength(&srr) ) THEN
              FOR rjumpindex _ 1 UP UNTIL > count DO
                 BEGIN
                 % advance through jump stack %
                    stid _ readsrring(&srr, rjumpindex : cc,
                    pvs1, pvs2);
                 %get the next statement%
                    % get statement length %
                       IF NOT lodprop( stid, txttyp : stdb)
                       THEN
                          err($"No text block associated with
                          node");
                       len _ [stdb].schars + 1; % number of
                       chars %
                    % form string of first 20 characters of
                    statement (or entire statement if
                    statement is < 20 characters long %
                       tp1 _ stid;
```

```
                        tp1[1] _ 1;
                        tp2 _ stid;
                        tp2[1] _ MIN(20,len);
                        IF len > 1 THEN *temp* _ tp1 tp2 ELSE
                        *temp* _ "<NULL>";
                     % append string to return list%
                        #rtnlst# !_ *temp*;
                  END
               ELSE err($"no entries in statement return ring");
               END;
            = 67: %- filereturn -%
               BEGIN
               IF (count _ frrlength(da.dalink)) THEN
                  FOR rjumpindex _ 1 UP UNTIL > count DO
                     BEGIN
                     %get the rjumpindex-th entry in the file
                     return ring and apperd file name to return
                     list%
                        INVOKE (badfrr);
                        filestraddr _ readfrring(da.dalink,
                        rjumpindex : &srr);
                        DROP (badfrr);
                        #rtnlst# !_ *[filestraddr]*;
                     END
               ELSE err($"no entries in file return ring");
               END;
            ENDCASE ABORT (badarg, $"The allowed values of the
            argument to xgetjumpring are RETURN (= 62) and
            FILERETURN (= 67).");
         % Create one list of string to return to FE %
            #rtnlst#[1] _ LIST( MOVE #rtnlst# );
      % Return %
         RETURN;
      (badfrr) CATCHPHRASE();                                    5V1E6
         BEGIN
         DISABLE (badfrr);
         CASE SIGNALTYPE OF
            = notetype :  NULL;
            = helptype :  NULL;
            = aborttype :
               IF SIGNAL = frremptyentry THEN
                  err($"No current entry in file return ring.");
            ENDCASE;
         CONTINUE;
         END;
      END.

   (xjmpcnt)   % CL: ;  Get search string form Jump command %
   PROCEDURE (searchtype REF, rtnlst REF );                      5V1F
      % Procedure description
         FUNCTION
            Get search string for Jump to Content/Word command.
            Return to FE for display via SHOW in grammar.
         ARGUMENTS
            searchtype:  command word variable
               type of search.  value -- content, word
```

```
            rtnlst:  list variable
               address of list for results
         RESULTS
            returned in rtnlst -- contents of search string or
            NULL if no previous search
         NON-STANDARD CONTROL
            go to "err" with msg "Not implemented yet" if invalid
            search type
         GLOBALS
            none
         %
      % Declarations %
         (locstr) REF;                                          5V1F2A
      % Set up return string %
         CASE ELEM #searchtype# [cwtype] OF
            = cwword, =cwcontent:
               CASE conreg.L OF
                  = 0 :  %no previous word / content %
                     #rtnlst# _ USE makedesc(unull, 0, FALSE);
                  ENDCASE
                     BEGIN  %put content/word in double quotes%
                     $locstr _ getstring(conreg.L + 2, $dspblk);
                     *locstr* _ """, *conreg*, """;
                     #rtnlst# _ *locstr*;
                     END;
            ENDCASE err($"Not implement yet -- xjmpcnt");
      % Return %
         RETURN;
      END.

%jump core routines%
   (cjump)   % U: jump to stid core routine %
   PROCEDURE (tp REF, vs REF, da REF);                          5V2A
      % Procedure description
         FUNCTION
            prepare the argumens for jump command
         ARGUMENTS
            tp -- REF -- pointer to character to be on top of
            screen (or CM)
            vs -- REF -- New viewspecs
            d -- REF display area referenced
         RESULTS
            none
         NON-STANDARD CONTROL
            none
         GLOBALS
            cspupdate, cspvs, curmkr
         %
      % set all the globals %
         cspupdate _ &da;
         cspvs _ vs;
         cspvs[1] _ vs[1];
         curmkr _ tp;
         curmkr[1] _ IF nlmode = fulldisplay THEN 1 ELSE tp[1];
         IF NOT nodisplay THEN dpset(dspjpf, curmkr, endfil,
         endfil);
```

```
        % Return %
          RETURN;
        END.

%load%
   %load PCP interface routine%
      (xload)                 % load a user program %
      PROCEDURE (fentity REF, windowid );                              5W1A
         LOCAL filnm REF;
         LOCAL adstr[40], retry;
         LOCAL STRING errstr[1999];
         cspupdate _ FALSE;
         &filnm _  BLEM#fentity#[tppair];
         % parse file name link %
            lnkprs( &filnm, $adstr);
         % load the program into the user programs buffer %
            cwndow _ windowid;
            IF cldprog( $adstr, TRUE %display messages%, $errstr)
            THEN
               err($errstr);
            IF errstr.L THEN typelit(1, $errstr);
         RETURN;
         END.

   %load core routines%
      %-NSW%                                                           5W2A
      (gpget)          % goto programs get file %
      PROCEDURE( adstr REF, msgflg );                                  5W2P
         % Procedure description
            FUNCTION
               Carried over from NLS8.5 to avoid changing code of
               old subsystems. use ldprog or cldprog
            ARGUMENTS
               See ldprog
            RESULTS
               none
            NON-STANDARD CONTROL
               none
            GLOBALS
               none
            %
         % call the loader %
            ldprog( &adstr, msgflg, 0 );
         RETURN;
         END.

      (cldprog)        % load program core routine %
      PROCEDURE( adstr REF, msgflg, errstr REF );                      5W2C
         % Procedure description
            FUNCTION
               This procedure interfaces to the loader by catching
               the appropriate signals.
            ARGUMENTS
               See ldprog
            RESULTS
               number of errors (0 - successful, 1 - failure)
```

```
            Number of undefined globals
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      LOCAL errors _ 0, strptr REF _ 0, udfnum _ 0;
      % invoke the catchphrase %
         INVOKE(badload, anycase);
      % call the loader %
         udfnum _ ldprog( &adstr, msgflg, &errstr );
      % come back here in any case %
         (anycase): DROP(badload);                                    5W2C5A
         IF &strptr AND &errstr THEN *errstr* _ *errstr*,
         *strptr*;
      RETURN( errors, udfnum );
      (badload) CATCHPHRASE(: &strptr);                               5W2C7
         BEGIN
         CASE SIGNALTYPE OF
            = aborttype :
               BEGIN
                  errors _ 1;
                  TERMINATE;
               END;
            ENDCASE;
         CONTINUE;
         END;
      END.

   (ldprog)       % Load a program into programs buffer %
   PROCEDURE( adstr REF, shomes, erstr REF );                          5W2D
      % Procedure description
         FUNCTION
            Invoke the TENLDR saved in our address space to load
            a rel file. The file is loaded into the user program
            buffer.
            It is not necessarry for the user to specify either
            the directory or the extension as this procedure will
            find it out.  It searches for the given filename with
            extensions: REL, CA, SK, SG, PROC-REP, SUBSYS, CGR in
            that order.  The search is repeated in the following
            directories (in this order): connected, login,
            default-programs.
            Use cldprog if you don't like signals
         ARGUMENTS
            adstr -- address of parsed link data structure
            shomes -- If TRUE, do dismes's
            erstr REF -- string to put loader output (undefined
            etc) or 0 for TTY window
         LOCALS
            savprcadr -- program buffer parameter saving
            savprogadr -- programs buffer parameter saving
            savsymloc -- symbol location saving
            jfn -- jfn of file to be loaded
            errs -- number of load errors (returned by loader).
            lc -- last loading location (returned by loader).
```

```
        loader -- pointing to the nls loader
        entry -- entry point to program just loaded
        ddtloc -- location of ddt pointer (of sym table)
        stname -- symbol table name of block
        i,j -- loop counting vars
        dirstr -- pointer to directory currently being
        searched
        ptype -- program type
        dirstack -- ordered list of directories to search (0
        means connected).
        extstack -- list of extensions in appropriate order.
        errstr -- error string
        dirname -- directory where the file was found.
        filename -- full length filename (no ^F/ESC)
        extname -- extension of the loaded file
        tempstr -- temporary use string.
    RESULTS
        Returns the number of undefined globals.  Does not
        return on erroneous loading.
    NON-STANDARD CONTROL
        Calls err or creates a signal if anything whatsoever
        goes wrong.
    GLOBALS
        symloc, upgskix, upgstk, upgnrs, ddtloc, ddmrkx,
        ddmrk, upgffbuf.
    %
LOCAL savprcadr, saveprogaddr, savsymloc, jfn, errs, lc,
loader REF, entry, ddtloc, stname, i, j, dirstr REF, ptype,
udfnum;
LOCAL dirstack _ (0, $userstr, $subdfdir);
LOCAL extstack _ ($relext, $caext, $skext, $sgext,
$procext, $subext, $cgrext);
LOCAL STRING errstr[199], dirname[39], filename[39],
extname[39], tempstr[199];
% get the file name specified by the parsed link data
structure %
   CASE lnbfls( 0, $adstr, $tempstr) OF
      = lhostn:  NULL;
      ENDCASE err($"Remote File Manipulations Not
      Implemented Yet");
% save away current state values %
   savsymloc _ symloc;
% get the jfn for the rel file %
   FOR i _ 0 UP UNTIL >= 3 DO
      BEGIN
         $dirstr _ dirstack[i];
         FOR j _ 0 UP UNTIL > 6 DO
            IF (jfn _ lgetjfn(&dirstr, $tempstr,
            extstack[j], 1B11, $errstr))
            THEN EXIT LOOP 2;
      END;
   IF NOT jfn THEN err($"File Not Found");
% get rid of any ^F or altmodes in filename %
   jfntostr( jfn, $dirname, 010000B6);
   jfntostr( jfn, $filename, 001000B6);
   jfntostr( jfn, $extname, 000100B6);
```

```
% find out what type of program for future instituting %
   ptype _ CASE TRUE OF
      = (*extname* = *relext*):  0;
      = (*extname* = *caext*):   1;
      = (*extname* = *skext*):   2;
      = (*extname* = *sgext*):   3;
      = (*extname* = *procext*): 4;
      = (*extname* = *cgrext*):  5;
      = (*extname* = *subext*):  6;
      ENDCASE 0;
% open the file %
   IF NOT SKIP !openf (jfn, 4400002B5) THEN
      BEGIN
      reljfn( jfn );
      err( $"Open File Failed" );
      END;
%adjust the user program buffer size if necessary to fit
the program being loaded%
   gpadjbsz(jfn);
% invoke the tenldr %
   IF shomes THEN dismes(1, $"Loading User Program");
   &loader _ nlsloader;
   errs _ loader(jfn, upgffbuf+1, upgbend, 0, &erstr :lc,
   udfnum);
IF errs THEN % not a good loading %
   BEGIN
      symloc _ savsymloc; % reset symloc as it may be
      clobbered by an error in loading %
      err( $"Error in Loading");
   END;
% mark the block in the user program buffer %
   IF upgskix >= $upgsksz THEN % no room in buffer %
      ABORT (upgerr, $"no more room in user program stack")
   ELSE % everything OK (so far) %
      BEGIN
      upgstk [upgskix _ upgskix + 1] _ upgffbuf;
      % set string into user program buffer %
         *upgnms* _ *upgnms*, SP, *filename*;
      END;
% update ddt's alt I-1 to include program just loaded %
   ddtloc _ ddtsptr; % get ptr to alt i -1 %
   [ddtloc] _ symloc; % stuff away new symloc %
% mark the block in the symbol table  %
   ddmrk[ddmrkx] _ savsymloc.RH - 1;
   IF (ddmrkx _ ddmrkx + 1 ) > ddmrkm THEN
      BEGIN
      BUMP DOWN ddmrkx;
      err($"Mark stack overflow");
      END;
% lookup file name and set entry instruction %
   stname _
      IF ptype = 5 %cgr% THEN
         mrkglookup( $filename : entry )
      ELSE
         ddtlookup( $filename, FALSE : entry );
   IF stname THEN %set up entry inst %
```

```
            [upgffbuf] _ 25489 + entry
         ELSE % entry not found %
            BEGIN
               CASE ptype OF
                  =0, =1, =2, =3: % REL, CA, SK, SG %
                     *tempstr* _ "WARNING -- no entry to program
                     ";
                  =4:                        % PROC-REP %
                     *tempstr* _ "WARNING -- No Procedure Named
                     ", *filename*;
                  =6:                        % SUBSYS %
                     ABORT(ermunknown, $"No dispatch table for
                     subsystem");
                  ENDCASE *tempstr* _ NULL;
               IF shomes THEN dismes(2,$tempstr);
            END;
      % set new limit on buffer space %
         saveprogaddr _ upgffbuf := lc;
      % institute program if so indicated by extension %
         *tempstr* _ NULL;
         CASE ptype OF
            = 1: % content analyzer %
               BEGIN
               *tempstr* _ "Instituting User Program as a Content
               Analyzer";
               [lda()].dacacode _ saveprogaddr;
               END;
            = 2: % sort key %
               BEGIN
               *tempstr* _ "Instituting User Program as a Sort
               Key Program";
               [lda()].daukeycod _ saveprogaddr;
               END;
            = 3: % sequence generator %
               BEGIN
               *tempstr* _ "Instituting User Program as a
               Sequence Generator";
               [lda()].dausqcod _ saveprogaddr;
               END;
            = 4: % procedure replace %
               IF stname THEN
                  BEGIN
                  IF rplproc( $filename, $filename) THEN
                     *tempstr* _ "Procedure ", *filename*, "
                     Replaced"
                  ELSE
                     *tempstr* _ "Old Procedure ", *filename*, "
                     Does Not Exist";
                  END;
            = 5:  % cgr %
               IF stname THEN
                  upgstk[upgskix].LH _ entry;
            = 6:  % subsys %
               % define the package just loaded%
                  upgstk[upgskix].LH _ crtlpkg( $filename, entry,
                  $dirname, 0, 0, 0);
```

```
            ENDCASE;
            IF shomes AND tempstr.L THEN dismes(2, $tempstr);
      RETURN( udfnum );
      END.

%-NSW%                                                              5W2E
%+NSW%                                                              5W2F
(gpget)    PROCEDURE   % Special NSW version that does Goto
Programs Get Rel File command %                                     5W2G
      (adstr,   % address of parsed link data structure %
      shomes);  % If TRUE, do dismes's; FALSE, only error mssages
      form loader get out. %
      % invoke the TENLDR saved in our address space to load a
      rel file
        the file is loaded into the user program buffer, and the
      tenldr expects three arguments set up %
      LOCAL savprcadr, saveprogaddr, savsymloc, jfn, errs, lc,
      loader, entry, ddtloc, oneflag, extflag, pgmfield, try,
      stname, i, ptype, dtstr[40];
      LOCAL TEXT POINTER tps,tpe, tpfe, tpx;
      LOCAL STRING errstr[200];
      LOCAL STRING recurse[200];
      LOCAL STRING locfilnam[100];
      LOCAL STRING filename[39];
      LOCAL STRING extname[39];
      LOCAL STRING tempstr[50];
      LOCAL STRING extension[10];
      LOCAL STRING str[200];
      REF adstr, loader;
      % initialize %
         oneflag _ TRUE;
      % get the file name specified by the parsed link data
      structure %
         CASE lnbfls( 0, &adstr, $str) OF
            = lhostn:  NULL;
            ENDCASE err($"Remote File Manipulations Not
            Implemented Yet");
      % edit filename string to get "extension" and filename-ext
      %
         FIND SF(*str*) ^tps SE(*str*) ^tpe;
         FIND ". ^tpe;  %remove trailing delimiter%
         IF FIND [".] ^tpfe > CH ^tpx THEN
            BEGIN
            extflag _ TRUE;
            *extname* _ +tpx tpe;
            *extension* _ tpx tpe;
            END;
         CASE TRUE OF
            = (*extname* = *relext*):  ptype _ 0;
            = (*extname* = *caext*):   ptype _ 1;
            = (*extname* = *skext*):   ptype _ 2;
            = (*extname* = *sgext*):   ptype _ 3;
            = (*extname* = *procext*): ptype _ 4;
            = (*extname* = *cgrext*):  ptype _ 5;
            = (*extname* = *subext*):  ptype _ 6;
            ENDCASE
```

```
            BEGIN
            ptype _ 0;
            *extension* _ "rel";
            tpfe _ tpe; %restore full name%
            tpfe[1] _ tpe[1];
            END;
         *filename* _ tos tpfe;
         *str* _ *filename*, ". ", *extension*;
      % save away current state values %
         savsymloc _ symloc;
      % get NSW file %
         IF NOT wmopen(0, $str, dontset, 0, $locfilnam, 0) THEN
            BEGIN
            *errstr* _ "Can't find ", *str*, " in NSW";
            err($errstr);
            END;
      % get jfn for local file %
         IF NOT jfn _ sgtjfn(1B5, $locfilnam, $errstr) THEN
         err($errstr);
      % open the file %
         IF NOT SKIP !openf (jfn, 440000285) THEN
            BEGIN
            IF NOT SKIP !rljfn( jfn ) THEN NULL;
            err( $"Open File Failed" );
            END;
      %adjust the user program buffer size if necessary to fit
      the program being loaded%
         gpadjbsz(jfn);
      % invoke the tenldr %
         IF shomes THEN dismes(1, $"Loading User Program");
         savprcadr _ ( prcadr := &nlsloader);
         errs _ sprocess(jfn,upgffbuf + 1, upgbend : lc);
         prcadr _ savprcadr;
         IF shomes THEN dismes(0);
      % delete local copy of file %
         IF NOT (R1 _ sgtjfn(0, $locfilnam, $errstr)) THEN
         err($errstr);
         IF NOT SKIP !JSYS delf THEN
            BEGIN
            *errstr* _ "Cannot delete local copy of program";
            err($errstr);
            END;
      % the tenldr returns an error count and the next address in
      the buffer %
         IF NOT errs THEN % good return %
            BEGIN
            % mark the block in the user program buffer %
               IF upgskix >= $upgsksz THEN
                  ABORT (upgerr, $"no more room in user program
                  stack")
               ELSE
                  % load was successful and there was room in the
                  buffer and there is room in stack %
                  BEGIN
                  upgstk [upgskix _ upgskix + 1] _ upgffbuf;
                  % set string into user program buffer %
```

```
            *upgnms* _ *upgnms*, SP, *filename*;
        END;
    % update ddt's alt I-1 to include program just loaded
    %
     ddtloc _ ddtsptr; % get ptr to alt i -1 %
     [ddtloc] _ symloc; % stuff away new symloc %
    % mark the block in the symbol table  %
     ddmrk[ddmrkx] _ savsymloc.RH - 1;
     IF (ddmrkx _ ddmrkx + 1 ) > ddmrkm THEN
        BEGIN
        ddmrkx _ ddmrkx - 1;
        err($"Mark stack overflow");
        END;
    % lookup file name and set entry instruction %
     stname _
        IF ptype = 5 %cgr% THEN
           mrkglookup( $filename : entry )
        ELSE
           ddtlookup( $filename, FALSE : entry );
     IF stname  THEN
        [upgffbuf] _ 254B9 + entry %set up entry inst %
     ELSE
        CASE ptype OF
           =0, =1, =2, =3:      % REL, CA, SK, SG %
              IF shomes THEN dismes(2,$"WARNING -- no
              entry to program ");
           =4:                  % PROC-REP %
              BEGIN
              *tempstr* _ "WARNING -- No Procedure
              Named ", *filename*;
              IF shomes THEN dismes(2,$tempstr);
              END;
           =5:                  % CGR %
              BEGIN
              *tempstr* _ "WARNING -- No Subsystem
              Named ", *filename*;
              IF shomes THEN dismes(2,$tempstr);
              END;
           =6:                  % SUBSYS %
              ABORT(ermunknown, $"No dispatch table for
              subsystem");
           ENDCASE;
    % set new limit on buffer space %
      saveprogaddr _ upgffbuf := lc;
    % institute program if so indicated by extension %
     CASE ptype OF
        = 1: % content analyzer %
           BEGIN
           IF shomes THEN dismes(2,$"Instituting User
           Program as a Content Analyzer");
           [lda()].dacacode _ saveprogaddr;
           END;
        = 2: % sort key %
           BEGIN
           IF shomes THEN dismes(2,$"Instituting User
           Program as a Sort Key Program");
```

```
                    [1da()].daukeycod _ saveprogaddr;
                    END;
                = 3: % sequence generator %
                    BEGIN
                    IF shomes THEN dismes(2,$"Instituting User
                    Program as a Sequence Generator");
                    [1da()].dausqcod _ saveprogaddr;
                    END;
                = 4: % procedure replace %
                    IF stname THEN
                       BEGIN
                       IF rplproc( $filename, $filename) THEN
                          *tempstr* _ "Procedure ", *filename*,
                          " Replaced"
                       ELSE
                          *tempstr* _ "Old Procedure ",
                          *filename*, " Does Not Exist";
                       IF shomes THEN dismes(2, $tempstr);
                       END;
                = 5:    % cgr %
                    IF stname THEN
                       BEGIN
                       upgstk[upgskix].LH _ entry;
                       % Noted in BEIGNORE
                       dfnsubsys( entry, gtctrlbits(entry),
                       $nlssubs );
                       dfnsubsys( entry, gtctrlbits(entry),
                       $allsubs );
                       %
                       *tempstr* _ "Subsystem ", *filename*, "
                       Now Available (Attached)";
                       IF shomes THEN dismes(2, $tempstr);
                       END;
                = 6:    % subsys %
                    BEGIN
                    *tempstr* _ *filename*, ".GRAM";
                    %possible that a cll to a FE parse function
                    to switch to a new grammar should go here%
                    % define the package just loaded%
                       upgstk[upgskix].LH _ crtlpkg( $filename,
                       entry,0);
                    END;
                ENDCASE;
            END
         ELSE
            BEGIN
            symloc _ savsymloc; % reset symloc as it may be
            clobbered by an error in loading %
            err( $"Error in Loading");
            END;
      RETURN;

      (nofront) CATCHPHRASE;                                    5W2G28
         BEGIN
         DISABLE(nofront);
         ABORT(err, $"Can't load frontend for this SUBSYS");
```

```
        END;

    END.

%+NSW%                                                          5W2H
(jutval)        % find value of name %                          5W2I
    PROCEDURE
        (name, mode);
    LOCAL retval, c;
    REF name;

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

%  Commented out for now -- needs to be fixed so that backend
doesn't try to change the frontend grammar

(gtctrlbits) PROCEDURE  %% return ctrl bits for new subsystem
%%                                                              5W2L
    (dptptr);                                                   5W2L1
    LOCAL instptr;
    REF dptptr, instptr;
    %% check to make sure that we've been given a valid pointer
    %%
        IF dptptr.dptvalid # $dptvldationcode THEN
            RETURN( FALSE );
    %% set up to go through rule nlssubs %%
        &instptr _ $nlssubs;
    %%if there are no defined subsystems then return 7%%
        IF instptr.opcode # $xecute THEN RETURN( 7 );
        &instptr _ instptr.addr;
    %% loop through rule %%
        WHILE &instptr DO
            IF instptr.opcode = $keyop THEN
                IF *[instptr.addr]* = *[dptptr.dptname]* THEN
                    RETURN( instptr.ctrl )  %% replacing existing
                    subsys %%
                ELSE IF *[instptr.addr]*[1] =
                *[dptptr.dptname]*[1] THEN
                    %% first char conflict %%
                        IF instptr.ctrl.llcmd THEN RETURN( 3 )
                        ELSE &instptr _ instptr.alternative
            ELSE &instptr _ instptr.alternative;
    %% not found so return all bits on %%
        RETURN( 7 );
    END.
```

```
%

(gpadjbsz) PROCEDURE % adjusts user program buffer size %  5W2P
   (jfn); %jfn of file which is going to be loaded%        5W2P1
   LOCAL
      wordsneeded,
      flength; %byte size of rel file, used as estimate of
      core size%
   % Estimate core size of the rel file%
      wordsneeded _ relfsize(jfn);
   IF NOT gpawdsleft(wordsneeded, FALSE) THEN
   err($"Insufficient room in user buffer; try deleting a
   program");
   RETURN (TRUE);
   END.
(gpawdsleft) PROCEDURE %try to adjust buffer size so there are
wordsneeded words left%                                    5W2Q
   (wordsneeded, %number of unused words to leave%
   downf); %adj. downward also%
   LOCAL
      wordsleft,  %in user program buffer%
      wordstoadd,
      pagestoadd,
      remainder,
      successf,
      newbsize;
   % Compute space left in user buffer %
      wordsleft _ upgbend - upgffbuf + 1;
   %adjust buffer size so it will fit (hopefully)%
      IF wordsneeded < wordsleft AND NOT downf THEN
      RETURN(TRUE);
      wordstoadd _ wordsneeded - wordsleft;
      pagestoadd _ (wordstoadd + 511) / 512;
      newbsize _ upgbsz + pagestoadd;
      successf _ TRUE;
      IF newbsize > maxbsize THEN
         BEGIN
         newbsize _ maxbsize;
         successf _ FALSE;
         END;
      gpbsz(newbsize);
      RETURN(successf);
   END.

(rplproc) % Replace procedure oldname by newname %          5W2R
   PROCEDURE
      (oldname, newname);
   LOCAL oldval, newval, c;
   REF oldname, newname;

   c _ 0;
   IF NOT (oldval _ jutval(&oldname, TRUE : c)) THEN
      err($"No old procedure: error in DDT lookup.");
   IF NOT (newval _ jutval(&newname, FALSE)) THEN
      err($"No new procedure: error in DDT lookup.");
   IF oldval = newval THEN RETURN( FALSE );
```

```
        % Put "old" procedure in backup table. %
           IF NOT c THEN
              IF NOT tblsearch($proctbl, 10 %proctblsz%, 2
              %proctblentsz%, 0: c) THEN
                 err($"Backup Table Full...Nothing Replaced");
           [c] _ oldval;
           [ c +1] _ [ oldval];
           [oldval] _ 254B9 %jrst% + newval;
        RETURN( TRUE );
        END.

%logout%
   %logout PCP interface routine%
      (xlogout)   % CL: ;  logout xroutine %
      PROCEDURE;                                                    5X1A
         % Procedure description
            FUNCTION
               Perform any cleanup tasks prior to the logout of the
               job associated with this session.
            ARGUMENTS
               none
            RESULTS
               proc-value
            NON-STANDARD CONTROL
               none
            GLOBALS
               none
            %
         % Declarations %
         clogout();
         % Return %
            RETURN;
         END.

   %logout core routines%
      (clogout)   % GB: core NLS Logout procedure %
      PROCEDURE %( => no value )%;                                  5X2A
         % Procedure description
            FUNCTION
               Logs out the user's job.
            ARGUMENTS
               none
            RESULTS
               none
            NON-STANDARD CONTROL
               none
            GLOBALS
               <lots>
            EXAMPLE
               clogout()
            %
         % Declarations %
         % see NLS 8.5 for code that deals with group login
         restrictions %
         RETURN;
         END.   %%
```

```
%mark%
   %mark PCP interface routine%
      (xmark) %Execute Mark Command%                                    5Y1A
         PROCEDURE (
            %FORMALS%
               destptr REF,  %pointer to char to be marked%
               mkrptr REF    %pointer to marker name%
               );
         LOCAL destination REF, marker REF;
         %----------------------%
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         IF &mkrptr THEN &marker _ ELEM #mkrptr#[tppair];
         cmarcha(&destination, &marker, &marker+d2sel);
         curmkr _ destination; curmkr[1] _ destination[1];
         RETURN;
         END.

   %mark core routines%
      (cmarcha)              %Core NLS Mark Character Command%   5Y2A
         %moves marker if pre-existant name%
         PROCEDURE (bug, tp1, tp2);
         LOCAL mname, count, length, stid, aring;
         LOCAL STRING astrng[25];
         REF bug, tp1, tp2, aring;
         % make sure we have a locked file %
            stid _ origin;
            stid.stfile _ bug.stfile;
            lodent( stid, rngtyp : &aring);
            aring.rsub _ aring.rsub;
         *astrng* _ tp1 tp2;
         count _ mname _ 0;
         length _ MIN (5, astrng.L);
         UNTIL (count _ count+1) > length DO
            CASE count OF
               = 1: mname.chr0 _ *astrng*[count];
               = 2: mname.chr1 _ *astrng*[count];
               = 3: mname.chr2 _ *astrng*[count];
               = 4: mname.chr3 _ *astrng*[count];
               = 5: mname.chr4 _ *astrng*[count];
               ENDCASE;
         insmkr (&bug, mname);
         RETURN;
         END.

%merge%
   % moved after FINISH -- never has worked %
%move%
   %move PCP interface routine%
      (xmove) %Execute Move Command%                                    5A@1A
         PROCEDURE (
            %FORMALS%
               stype REF,    %source entity type%
               sourceptr REF,        %source pointer%
               dtype REF,    %destination entity type%
               destptr REF,  %destination pointer%
```

```
            level,                  %level adjustment string%
            vs REF,                 %viewspec string%
            txtrtn,          %if TRUE, return text of statement%
            tprtn            %if TRUE, return text pointer(s)%
            );
         LOCAL
            source REF, destination REF, endstid, grplist REF,
            retry REF, i, savedest,
            type, sourceda REF, destda REF, da REF, x, y, r,
            rhostr, rhost2, tlngth, adstr[40];
         LOCAL STRING
            filstr[200], filst2[200], badmsg[200];
         LOCAL LIST tplist [6], txtlist[2], restore[2];
      %----------------------%
      &source _ ELEM #sourceptr#[tppair];
      IF &sourceptr THEN &source _ ELEM #sourceptr#[tppair];
      IF &destptr THEN &destination _ ELEM #destptr#[tppair];
      CASE ELEM #stype#[cwtype] OF
         = 30 %- link -%:
            BEGIN
            lnkprs( &source, $adstr);
            source _ adstr[ls];
               source[1] _ adstr[ls+1];
            [&source+d2sel] _ adstr[le];
               [&source+d2sel+1] _ adstr[le+1];
            END;
         ENDCASE;
      CASE ELEM #stype#[cwtype] OF
         = 2 %- character -%, = 11 %- invisible -%, = 1 %- text
         -%:
            BEGIN
            IF NOT nortnrings THEN
               clist(ctcmk, destination.stfile, source.stfile);
            IF NOT nodisplay THEN
               dpset(dsprfmt, destination, source, endfil);
            curmkr _ destination[d2sel];
               curmkr[1] _ destination[d2sel+1] +
               source[d2sel+1]-source[1]-1;
            restore _ source; restore[1] _ source[1] - 1;
            IF NOT source.stastr THEN
               cmovtex(&destination+d2sel, &source,
               &source+d2sel, FALSE)
            ELSE
               cinstex(&destination+d2sel, &source,
               &source+d2sel, FALSE);
            destination[d2sel+1] _ destination[d2sel+1] + 1;
            %For returning handles to ISI%
            IF NOT nortnrings THEN clupdt();
            IF txtrtn THEN
               BEGIN
               #txtlist# _ source;
               IF NOT source = curmkr THEN #txtlist# !_ curmkr;
               END;
            END;
         = 3 %- word -%, = 4 %- visible -%, = 8 %- integer -%, =
         30 %- link -%:
```

```
            BEGIN
            IF NOT nortnrings THEN
               clist(ctcmk, destination.stfile, source.stfile);
            IF NOT nodisplay THEN
               dpset(dsprfmt, destination, source, endfil);
            curmkr _ destination[d2sel];
               curmkr[1] _ destination[d2sel+1] +
               source[d2sel+1]-source[1];
            restore _ source; restore[1] _ source[1] - 1;
            IF NOT source.stastr THEN
               cmovtex(&destination+d2sel, &source,
               &source+d2sel, TRUE)
            ELSE
               cinstex(&destination+d2sel, &source,
               &source+d2sel, TRUE);
            destination[d2sel+1] _ destination[d2sel+1] + 1;
            %For returning handles to ISI%
            IF NOT nortnrings THEN clupdt();
            IF txtrtn THEN
               BEGIN
               #txtlist# _ source;
               IF NOT source = curmkr THEN #txtlist# !_ curmkr;
               END;
            END;
         = 29 %- statement -%:
            BEGIN
            curmkr _ xcmst(&destination, level, &source,
            moveflag, &vs);
            curmkr[1] _ 1;
            IF txtrtn THEN #txtlist# _ curmkr;
            destination[d2sel] _ curmkr;  %For returning handles
            to ISI%
            destination[d2sel + 1] _ 0;
            END;
         = 27 %- group -%, = 28 %- plex -%, = 26 %- branch -%:
            BEGIN
            curmkr _ xcmgrp(&destination, level, &source,
            moveflag, &vs);
            curmkr[1] _ 1;
            destination[d2sel +1] _ 0;
            CASE level OF
               = 0: destination[d2sel] _
               getsuc(destination[d2sel]);
               = -1: destination[d2sel] _
               getsuc(destination[d2sel]);
            ENDCASE
               BEGIN
               FOR i _ 1 UP UNTIL > level DO
                  destination[d2sel] _ getup(destination[d2sel]);
               destination[d2sel] _ getsuc(destination[d2sel]);
               END;
            IF txtrtn THEN
               BEGIN
               &grplist _ alloclist(20);
               endstid _ getend(curmkr);
               #grplist# _ savedest _ destination[d2sel];
```

```
                DO
                   BEGIN
                   #grplist# !_ destination[d2sel] _
                   getnxt(destination);
                   IF grplist.L = grplist.M THEN
                      &grplist _ alloclist(grplist.M + 20);
                   END
                UNTIL destination[d2sel] = endstid;
                destination[d2sel] _ savedest;
                END;
             END;
          ENDCASE
          IF NOT donthelp THEN
             BEGIN
             *badmsg* _ "The selection type was incorrect.  The
             allowed values are Branch, Character, Group, ";
             *badmsg* _ *badmsg*, "Integer, Invisible, Link, Plex,
             Statement, Text, Visible, or Word.";
             &retry _ HELP (badarg, $badmsg, cindex);
             source _ ELEM #retry#[tppair];
             REPEAT CASE (ELEM #retry#[cwtype]);
             END
          ELSE ABORT (badarg, $"The selection type was
          incorrect.");
       %
       IF tprtn THEN
          BEGIN
          #tplist# _ restore, restore[1];
          #tplist# !_ destination[d2sel], destination[d2sel+1];
          IF NOT ((destination[d2sel] = curmkr) AND
          (destination[d2sel+1] = curmkr[1])) THEN
             #tplist# !_ curmkr, curmkr[1];
          END;
       IF tprtn THEN
          IF txtrtn THEN RETURN(grplist, tplist)
          ELSE RETURN
       ELSE
          IF txtrtn THEN RETURN(grplist, empty)
          ELSE RETURN;
       %
       RETURN;
       END.

%move core routines%
   (cmovgro)                %Core NLs Move Group Command%      5A02A
      PROCEDURE (tostid, rlevcnt, fromstid1, fromstid2, vsptr);
      LOCAL newhead, newtail, work;
      REF vsptr;

      fromstid1 _ grptst(fromstid1, fromstid2 : fromstid2);
      movtst(tostid, fromstid1, fromstid2);
      IF tostid.stfile = fromstid1.stfile THEN
         BEGIN %intrafile move%
         IF NOT &vsptr THEN
            BEGIN
            IF NOT remgrp(fromstid1, fromstid2) THEN
```

```
            ABORT(badmove,$"Illegal move");
            insgrp(tostid, rlevcnt, fromstid1, fromstid2);
            END
         ELSE
            mvdlfgrp( fromstid1, fromstid2, &vsptr,
               IF (work _ getnxt(getend(fromstid2))) # endfil
               THEN work ELSE getbck(fromstid1), moveflag,
               tostid, rlevcnt);
         newhead _ fromstid1;  newtail _ fromstid2;
         END
      ELSE
         BEGIN %cross file move%
         newhead _
            ccopgro(tostid, rlevcnt, fromstid1, fromstid2,
            &vsptr: newtail);
         cdelgro(fromstid1, fromstid2, &vsptr);
         END;
      RETURN(newhead, newtail);
      END.

(movtst)                                                    5A@2B
      %Given an stid as the first argument, this routine makes
      sure that the stid is not in the group bounded by the
      second and third arguments passed it.  If it is, it does an
      ABORT.
         First it sees whether STID is a suc of GRP2 on the same
         level.  If not, it then sees whether STID's sucs (and
         ups) are GRP1 or GRP2.%
      %-------------%
      PROCEDURE(stid, grp1, grp2);
      LOCAL tmpsid; %working stid%
      IF grp2.stpsid = origin THEN ABORT(badmove,$"Illegal
      move");
      IF grp1.stfile # grp2.stfile THEN ABORT(badgrp,$"Illegal
      group");
      IF stid.stfile # grp1.stfile
         OR stid.stpsid = origin THEN RETURN;
      IF stid = grp1 OR stid = grp2 THEN ABORT(badmove,$"Illegal
      move");
      tmpsid _ grp2;
      LOOP %is stid in plex after grp2%
         BEGIN
         IF getftl(tmpsid) THEN
            BEGIN
            tmpsid _ getsuc(tmpsid);
            EXIT;
            END;
         tmpsid _ getsuc(tmpsid);
         IF stid = tmpsid THEN RETURN; %psid in plex after grp2%
         END;
      LOOP %see if grp1, grp2 in superstructure of psid%
         BEGIN
         IF stid = grp1 THEN ABORT(badmove,$"Illegal move");
         IF stid.stpsid = origin OR stid = tmpsid THEN RETURN;
         LOOP
            BEGIN
```

```
            IF stid = grp2 THEN ABORT(badmove,$"Illegal move");
            IF getftl(stid) THEN EXIT;
            stid _ getsuc(stid);
            IF stid = grp1 THEN RETURN;
            END;
         stid _ getsuc(stid);
         END;
      END.

(cmovsta)                %Core NLS Move Statement Command%  5A@2C
   PROCEDURE (tostid, rlevcnt, fromstid, vsptr);
   REF vsptr;
   IF getsub(fromstid) # fromstid THEN
      ABORT(badmove,$"Illegal move");
   tostid _ cmovgro(tostid, rlevcnt, fromstid, fromstid,
   &vsptr);
   RETURN(tostid);
   END.

(cmovtex)                %Core NLS Move Text Command%       5A@2D
   PROCEDURE (totptr, fromtptr1, fromtptr2, insertspace);
   LOCAL TEXT POINTER del1, del2;
   LOCAL STRING locstr[5];
   REF totptr, fromtptr1, fromtptr2;
     %msntx();%
   del1 _ fromtptr1;  del1[1] _ fromtptr1[1];
   del2 _ fromtptr2;  del2[1] _ fromtptr2[1];
   IF insertspace THEN
      BEGIN
      *locstr* _ SP;
      incspc($del1, $del2);
      END
   ELSE *locstr* _ NULL;
   IF POS SF(totptr) = SF(fromtptr1) THEN
      IF POS totptr < fromtptr1 THEN
         ST totptr _ SF(totptr) totptr, *locstr*, fromtptr1
         fromtptr2,
            totptr del1, del2 SE(totptr)
      ELSE
         ST totptr _ SF(totptr) del1, del2 totptr,
            *locstr*, fromtptr1 fromtptr2, totptr SE(totptr)
   ELSE
      BEGIN
      ST totptr _ SF(totptr) totptr,
         *locstr*, fromtptr1 fromtptr2, totptr SE(totptr);
      ST fromtptr1 _ SF(fromtptr1) del1, del2 SE(fromtptr1)
      END;
     %msftx();%
   RETURN;
   END.

%open%
   %open PCP interface routine%
      (xopen) %Execute Open Command%                          5AA1A
         PROCEDURE (
            %FORMALS%
```

```
            fileptr REF,  %name of file to be opened%
            autoclose,    %if TRUE, close automatically%
            access,       %access to be given file%
            window        %number of window to display it in%
            );
         LOCAL hostno, da REF, fileno, stid, pcap, tp REF,
         filename REF;
         LOCAL STRING hnmono[10], %host name or number%
            filestr[200], locflnm[200];
         LOCAL TEXT POINTER ptr, ptr2;
      %----------------------%
      &da _ cspupdate _ dsparea(window);
      IF &fileptr THEN &filename _ ELEM #fileptr#[tppair];
      &tp _ &filename+d2sel;
      % move file name to local string %
         CASE hostno _ lnbfls( &filename, 0, $filestr) OF
            = lhostn: NULL;  %local host%
            = sriai20host, = sriaikahost, = isichost, = isiehost:
            %valid NLS hosts%
               BEGIN
               dismes(1,$"Loading Remote File");
               IF NOT FIND SE(*filestr*) ['.]
                  THEN *filestr* _ *filestr*, ".NLS";
               IF FIND SF(*filestr*) [">] ^ptr ['.] ^ptr2 _ ptr2
                  THEN *locflnm* _ ptr ptr2, "-REM", ptr2
                  SE(*filestr*);
               *hnmono* _ STRING(hostno);
               fetchfile($hnmono, $locflnm, $filestr);
               *filestr* _ *locflnm*;
               dismes(1,$"Edits do not appear in remote files.");
               dismes(1,$"Stored locally as");
               END;
            ENDCASE
               dismes(1, $"Not a valid NLS host");
      IF (fileno _ cloafil($filestr)) THEN
         BEGIN
         IF NOT autoclose THEN [flntadr(fileno)].flnoclose _
         TRUE;   %don't close automatically%
         IF access THEN setaccess(fileno, access);   %NLS access%
         curmkr _ orgstid;
         curmkr.stfile _ fileno;
         curmkr[1] _ 1;
         END;
      cspvs _ da.davspec;
      cspvs[1] _ da.davspc2;
      IF NOT nodisplay THEN dpset(dspyes, curmkr, endfil,
      endfil);
      RETURN;
      END.

   %open core routines%
      (cloafil) PROCEDURE   % Core NLS Load File command %      5AA2A
         (flnam);  % string containing name of file to load %  5AA2A1
         % cloafil gets a jfn for the file and calls OPEN.  It
         returns the NLS file number of the file. %
         %----------------%
```

```
        LOCAL fileno, jfn = 0, sysmsg;
        REF flnam;
        %-NSW%                                                    5AA2A6
        % Get jfn (using long gtjfn) %
           IF NOT jfn _ lgetjfn (0, &flnam, $nlsext, gtjoif, $lit)
           THEN
              ABORT (ofilerr, $lit);
        INVOKE (sigsig);
        %open the file if not now open--allocates a new file number
        if necessary--open pc if there is one%
           fileno _ open (jfn, &flnam);
        %-NSW%                                                    5AA2A10
        %+NSW%                                                    5AA2A11
           INVOKE (sigsig);
           %open the file if not now open--allocates a new file
           number if necessary--open pc if there is one%
              fileno _ open (jfn, &flnam, 0);  %unknown sem value%
        %+NSW%                                                    5AA2A12
        DROP (sigsig);
        RETURN (fileno);

        (sigsig) CATCHPHRASE(:sysmsg);                            5AA2A16
           CASE SIGNAL OF
              = errsig:
                 BEGIN
                 DISABLE (sigsig);
                 ABORT (ofilerr, sysmsg);
                 CONTINUE;
                 END;
              ENDCASE
                 BEGIN
                 DISABLE (sigsig);
                 CONTINUE;
                 END;

        END.

%output%
   %output PCP interface routine%
      (xoutput) %Execute Output Command% PROCEDURE (              5AB1A
         %FORMALS%
            entity REF,                 %entity type%
            fnameptr REF,     %file name pointer%
            window,           %number of last window bugged%
            param1 REF,                %&parameters%
            param2 REF,
            param3 REF,
            param4 REF                 % com machine type%
            );
         %LOCALS%
            LOCAL retry, tp REF, filename REF, startloc REF,
            opflags, devtype, entint, emsg;
            LOCAL STRING tipstr[10], locstr[200], outfile[30],
            badmsg[150];
         %----------------------%
         INVOKE (outfail, outreturn);
```

```
        $filename _ IF &fnameptr THEN  ELEM #fnameptr#[tppair] ELSE
        0;
        CASE entint _ ELEM #entity#[cwtype] OF
           = 51 %- quickprint -%, = 52 %- sequential -%, = 53 %-
           journal -%, = 54 %-printer -%, = 55 %- com -%, = 56
           %-assembler -%:
              BEGIN
              % decode number of copies string if necessary %
                 IF NOT &param1 THEN &param1 _ 1;
              % get output file name to locstr, use dfltname to
              construct the name if none was specified %
                 IF &filename THEN % use name supplied by user %
                    BEGIN
                    % filename is the address of an array of two
                    text pointers delimiting a file name string.
                    It is NOT the address of the string itself! %
                    IF FIND SF(filename) ([*lpt:*] / [*LPT:*] )
                    THEN *locstr* _ *LPT:*
                    ELSE CASE lnbfls( &filename, 0, $locstr) OF
                       = lhostn: NULL;
                       ENDCASE
                          ABORT(noremote, $*Remote File
                          Manipulations Not Implemented Yet*)
                    END
                 ELSE % construct default name %
                    dfltname(window, $locstr, entint, &param3 %
                    checked only if type = COM: test or not; will
                    append *COM* as extension to file name % );
              CASE entint OF
                 = 51 %- quickprint -%:
                    BEGIN
                    IF NOT FIND SF(*locstr*) [*.*] THEN
                       *locstr* _ *locstr*, *.*, STRING(&param1);
                          %make ext be number of copies%
                    coutgui($locstr, dsparea(window), &param3
                    %print headers%, &param2 %append to seqfile%);
                    END;
                 = 52 %- sequential -%:
                    BEGIN
                    coutseqfil($locstr, dsparea(window) , &param2
                    %append to seqfile%, &param3 % force upper
                    case? %);
                    END;
                 = 53 %- journal -%: %submission form%
                    BEGIN
                    IF NOT FIND SF(*locstr*) [*.*] THEN
                       *locstr* _ *locstr*, *.*, STRING(&param1);
                          %make ext be number of copies%
                    coutjouqui($locstr, dsparea(window), &param3
                    %print headers%, &param2 %append to seqfile%);
                    END;
                 = 54 %- printer -%:
                    BEGIN
                    IF NOT FIND SF(*locstr*) [*.*] THEN
                       *locstr* _ *locstr*, *.*, STRING(&param1);
                          %make ext be number of copies%
```

```
                    coutproc($locstr, dsparea(window), &param2
                    %append to seqfile%, opprdv % device type:
                    printer/printer file %, 0 % Not a com device %,
                    0 %opflags%);
                    END;
                = 55 %- com -%:    %Computer output to Microfilm%
                    BEGIN
                    IF NOT FIND SF(*locstr*) ['.] THEN
                       *locstr* _ *locstr*, '.', STRING(&param1);
                          %make ext be number of copies%
                    coutproc($locstr, dsparea(window), &param2
                    %append to seqfile%,  IF &param3 THEN opxpdv %
                    COM test % ELSE opcmdv, ELEM#param4#[cwtype]  %
                    COM device type %, 0 % opflags %);
                    END;
                = 56 %- assembler -%:
                    BEGIN
                    coutassfil($locstr, dsparea(window), &param2
                    %append to seqfile%, &param3 % force upper case
                    %);
                    END;
                ENDCASE;
            END;
        = 57 %- terminal -%, = 58 %- remote -%, = 59 %- comtest
        terminal -%:
            BEGIN
            % setup flags record (to be passed to OP) %
                opflags _ 0;
                opflags.opdisflg _ (nlmode = fulldisplay);
                CASE &param2 %send formfeeds% OF
                    = 1:
                        BEGIN
                        opflags.opform _ TRUE;
                        %opflags.opsimff _ FALSE;%
                        END;
                    ENDCASE
                        BEGIN
                        %opflags.opform _ FALSE;%
                        CASE &param3 %simulate formfeeds% OF
                           = 1:                opflags.opsimff _ TRUE;
                           ENDCASE   %opflags.opsimff _ FALSE%;
                        END;
                CASE &param4 %wait at page breaks% OF
                    = 1:    opflags.opwtpb _ TRUE;
                    ENDCASE %opflags.opwtpb _ FALSE%;
            % construct file name %
                CASE entint OF
                    = 57 %- terminal -%:
                        BEGIN
                        IF NOT &filename THEN
                           BEGIN
                           % Output to local terminal %
                           outfile.L _ 0;
                           devtype _ optydv;
                           END
                        ELSE
```

```
                        BEGIN
                        % Output terminal file %
                        CASE lnbfls( &filename, 0, $outfile) OF
                           = lhostn: NULL;
                           ENDCASE
                              ABORT(noremote, $"Remote File
                              Manipulations Not Implemented
                              Yet");
                        devtype _ optyfl;
                        END;
                     END;
                  = 58 %- remote -%: %printer/terminal%
                     BEGIN
                     devtype _ oprmdv;
                     &tp _ filename + d2sel; % filename has tip
                     number (in string form) %
                     *tipstr* _ filename tp;
                     *outfile* _ "NET:0.",
                        STRING(VALUE($tipstr), 8), "-,
                         STRING((&param1* 65536 + 2), 8);
                     END;
                  = 59 % comtest terminal %:
                     BEGIN
                     devtype _ opxtdv;
                     outfile.L _ 0;
                     END;
                  ENDCASE;
            coutproc($outfile, dsparea(window), FALSE % don't
            append %, devtype, IF entint=59 %- comtest terminal
            -% THEN ELEM#param4#[cwtype] % COM device type % ELSE
             0 % not a com device %, opflags);
            END;
         ENDCASE
            IF NOT donthelp THEN
               BEGIN
               *badmsg* _ "The type argument was incorrect.  The
               allowed values are Assembler, ";
               *badmsg* _ *badmsg*, "Com, Journal, Printer,
               Quickprint, Remote, Sequential, or Terminal.";
               retry _ HELP (badarg, $badmsg, cindex);
               REPEAT CASE (retry);
               END
            ELSE ABORT (badarg, $"The type argument was
            incorrect.");
      (outreturn):                                              5AB1A7
         DROP (outfail);
         RETURN;
      (outfail) CATCHPHRASE(:emsg);                             5AB1A8
         BEGIN
         CASE SIGNALTYPE OF
            =aborttype:
               BEGIN
               DISABLE (outfail);
               dismes(1, emsg);
               TERMINATE;
               END;
```

```
            ENDCASE
               CONTINUE;
         END;
      END.

%output support routines%
   (dfltname) % construct default output file name %          5AB2A
      PROCEDURE(
      % FORMAL ARGUMENTS %
         window,  %number of last window bugged%
         str,     % ptr to result astring %
         type,    % $quickprint, $com, etc. %
         tstpar); % $test OR NULL %
      LOCAL TEXT POINTER tp1, tp2;
      REF % VARIABLES %
         str;
      % ------------------ %
      %put file name into a string%
         *str* _ NULL;
         filnam (Edsparea(window)].dacsp.stfile, &str);
      % Check for NSW %
         IF nswsw THEN   %in NSW --no directories%
            BEGIN
            % check file name %
               IF NOT (FIND SF(*str*) ^tp1 ['.] < CH ^tp2) THEN
                  ABORT (badfname, $"Invalid default file name
                  constructed.");
            % edit string, user's initials into it %
               *str* _ '-, *initsr*, '-, tp1 tp2;
               IF type = 105 %+ com +% AND NOT tstpar THEN
                  *str* _ *str*, ".COM";
               END
            ELSE
               BEGIN  %not in NSW%
               % check file name %
                  IF NOT (FIND SF(*str*) ['',] $SP ^tp1 ['.] < CH
                  ^tp2) THEN
                     ABORT (badfname, $"Invalid default file name
                     constructed.");
               % edit string, putting printer directory and
               user's initials into it %
                  *str* _ '$, *initsr*, '$, tp1 tp2;
                  IF type = 105 %+ com +% AND NOT tstpar THEN
                     *str* _ "<COM>", *str*, ".COM"
                  ELSE *str* _ '<,  *prtdir*, '>, *str*;
               END;
      RETURN;
      END.

%output core routines%
   (coutassfil)   %Core NLS Output to Assembler file command %
                                                          5AB3A
      PROCEDURE (ofilnam, da, appendflag, forceup);
      REF ofilnam, da;
      ogphfg _ TRUE; % no headers %
      ogapfg _ appendflag;
```

```
    outasm (&ofilnam, &da, forceup);
    RETURN;
    END.

(coutjouqui)   %Core NLS Output Journal Quickprint command %
                                                          5AB3B
    PROCEDURE (ofilnam, da, noheadflag, appendflag);
    REF ofilnam, da;
    ogmhfg _ noheadflag;
    ogapfg _ appendflag;
    outjm (&ofilnam, &da);
    RETURN;
    END.

(coutqui)              %Core NLS Output Quickprint command %
                                                          5AB3C
    PROCEDURE (ofilnam, da, noheadflag, appendflag);
    REF ofilnam, da;
    ogmhfg _ noheadflag;
    ogapfg _ appendflag;
    outqp (&ofilnam, &da);
    RETURN;
    END.

(coutseqfil)   %Core NLS Output Sequential File command % 5AB3D
    PROCEDURE (ofilnam, da, appendflag, forceup);
    REF ofilnam, da;
    ogmhfg _ TRUE; % No headers %
    ogapfg _ appendflag;
    outseq (&ofilnam, &da, forceup);
    RETURN;
    END.

(coutproc)             %Core NLS Invoke Output Processor %
                                                          5AB3E
    PROCEDURE
    (ofilnam,    % string containing the name of the output file
    %
    da,          % display area descriptor %
    appendflag,  % flag - append to end of seq file %
    device,      % which device %
    comdevice,   % which phototypesetter:
       51 = SINGER, 50 = COMP80, 52 = 800VIDEOCOMP, 53 =
       500VIDEOCOMP %
    opflags);    % device dependant flags for OP %

    LOCAL jfn, errors, prcjfn, opentype, adstr[40];
    LOCAL gproc;  %graphics procedure called by O.P. for COM %
    LOCAL TEXT POINTER tp;
    LOCAL STRING gname[20];  %name of graphics user program%
    REF ofilnam, da;

    % nonsense globals (in this case!) %
       ogmhfg _ TRUE; % No special "quickprint" headers. %
       ogapfg _ appendflag;
       jfn _ prcjfn _ 0;
```

```
        % open the output file %
           % get jfn and open output file (or device for remote
           terminal); not done for local terminal %
              IF device # optydv AND device # opxtdv THEN
                 BEGIN %not output local terminal or comtest local
                 terminal %
                 IF (NOT oqapfg) OR ( FIND SF(*ofilnam*) "<
                 *prtdir* "> ) THEN
                    errors _ gtjoosf
                       % default to next version higher  if not
                       append to file or if append, but directory
                       is the printer directory %
                 ELSE errors _ 0; % use highest existing version %
                 IF NOT jfn _ lgetjfn (0, &ofilnam, $txtext,
                 errors, $lit) THEN
                    ABORT (ofilerr, $lit);  %Previously a signal%
                 INVOKE(jfnclose);
                 IF device = oprmdv THEN
                    oqapfg _  FALSE
                 ELSE
                    BEGIN
                    !gtfdb( jfn, 1B6+1, $R3);
                    IF oqapfg THEN
                       oqapfg _ IF R3.fdbnxf THEN FALSE ELSE TRUE;
                    END;
                 CASE device OF
                    = opcmdv: % com % opentype _ comtyp;
                    = oprmdv: % remote terminal % opentype _
                    netype;
                    ENDCASE % printer, terminal file % opentype _
                    chrtyp;
                    % NOTE: comtyp and chrtyp result in the same
                    things happening: namely, 7 bit bytes specified
                    to sysopen.  Netyp results in 8 bit bytes and
                    "buffered send mode" as explained in JSYS
                    manual %
                 IF NOT sysopen (jfn, IF oqapfg THEN append ELSE
                 write, opentype, $lit) THEN
                    BEGIN
                    reljfn (jfn:=0);
                    DROP(jfnclose);
                    ABORT (ofilerr, $lit);  %Previously a signal%
                    END;
                 END
              ELSE jfn _ oqapfg _  FALSE;
        % get graphics output address if COM %
           IF device # opcmdv THEN gproc _ 0
           ELSE
              BEGIN
              % Get address of GCOUT (diagram producing procedure);
              if it isn't there, we must load the graphics system
              for diagram formatting.  %
                 IF NOT ddtlookup($"GCOUT", FALSE, % get procedure
                 in user program. %: gproc) THEN
                    BEGIN
                    % Must load graphics program %
```

```
                % Increase buffer size to 30 pages. %
                   IF NOT gpbsz( 30 ) THEN
                      err($"Cannot set user program buffer for
                      graphics");
                % Reset the user program buffer. %
                   gpgmrst();
                   dismes(2,$"all user programs deleted except
                   graphics");
                % Load the user program with GRAPHICS code. %
                   *gname* _ "<graphics,>"; %ldprog knows
                   directory%
                   FIND SF(*gname*) ^tp;
                   INVOKE (sigload);
                   lnkprs($tp, $adstr);
                   ldprog($adstr, FALSE %don't print messages%,
                   0);
                   DROP (sigload);
                % Get address of GCOUT (diagram producing
                procedure) %
                   IF NOT ddtlookup($"GCOUT", FALSE, % get
                   procedure in user program. %: gproc) THEN
                   err($"Cannot find gcout address");
                END;
          END;
    opinit (&da, jfn % 0 for local terminal outputs %, device,
    opflags, gproc, comdevice);
    tp[0] _ da.dacsp; tp[1] _ 1;
    IF NOT (prcjfn _ sgtjfn(gtjprf, IF exp THEN $xopname ELSE
    $opname, $lit)) THEN
       ABORT (prcerr, $lit);    %Previously a signal%
    errors _ processor
       (0, $tp, &da, odtyp, jfn, prcjfn);
    % close output file %
       IF jfn  THEN
          IF NOT sysclose (jfn, $lit) THEN
             BEGIN
             reljfn (jfn:=0);
             DROP(jfnclose);
             ABORT (prcerr, $lit);      %Previously a signal%
             END
          ELSE jfn _ 0;
       DROP(jfnclose);
    % if waiting at page breaks, wait before returning
       IF opflags.opwtpb THEN DO
          jfn _ inpt()
       UNTIL jfn # LF;
    %
    RETURN;
    (sigload) CATCHPHRASE;                                  5AB3E24
       BEGIN
       DISABLE (sigload);
       ABORT(err,$"Cannot load GRAPHICS programs");
       CONTINUE;
       END;
    (jfnclose) CATCHPHRASE;                                 5AB3E25
       BEGIN
```

```
            DISABLE (jfnclose);
            IF jfn THEN
               IF NOT sysclose(jfn, $lit) THEN reljfn(jfn:=0);
            CONTINUE;
            END;
         END.

%quit%
   %quit PCP interface routine%
      (xquitsubsys) PROCEDURE; %quit current subsystem%          5AC1A
         clopkg(curpkg);
         curpkg _ prvpkg;
         RETURN;
         END.

FINISH of psedit1
% Obsolete JUMP command code %
   (xjrinit) PROCEDURE %initilize for jump return and jump file return%
                                                                    8A
      % FORMAL ARGUMENTS %
         (entity REF);  %type of jump%
      % -------------- %
      CASE ELEM #entity#[cwtype] OF
         = 62 %- return -%: srrcnt _ 0;
         = 67 %- filereturn -%: frrcnt _ 0;
         ENDCASE ABORT (badarg, $"The allowed valued of the argument to
         xjrinit are RETURN (= 62) and FILERETURN (= 67).");
      rjumpindex _ 0;  %Init the index of ring entries%
      RETURN;
      END.

   (xringjump) PROCEDURE( % provides feedback for stepping through the
   rings for jump RETURN and jump FILE RETURN %                     8B
      % FORMAL ARGUMENTS %
         entity REF,     % entity type for the jump %
         window,         %number of the window to jump in%
         rtnlist REF);  %return arg%
      LOCAL da REF, stid, stdb, len, cc, pvs1, pvs2, srr REF,
      filestraddr, retry;
      LOCAL TEXT POINTER tp1,tp2;
      LOCAL STRING temp[200];
      LOCAL LIST locres[6];
      % -------------- %
      %set da to current display area%
         &da _ dsparea(window);
      %update the index of ring entries%
         BUMP rjumpindex;
      CASE ELEM #entity#[cwtype] OF
         = 62 %- return -%  %statement return ring%:
            BEGIN
            % advance through jump stack %
               readfrring(da.dalink, 0 : &srr);
               stid _ readsrring(&srr, rjumpindex : cc, pvs1, pvs2);
            %return the current statement%
               % get statement length %
                  IF NOT lodprop( stid, txttyp : stdb) THEN
```

```
                err($"No text block associated with node");
             len _ [stdbl.schars + 1; % number of chars %
          % construct text ptrs to each end of stmt %
             tp1 _ stid;
             tp1[1] _ 1;
             tp2 _ stid;
             tp2[1] _ len;
          % assign the statement to the string "temp" %
             IF len > 1 THEN *temp* _ tp1 tp2 ELSE *temp* _
             "<NULL>";
             dismes (2, $temp);
       % save the current state in the return list %
          #locresl#[1] _ stid;
          #locresl#[2] _ cc;
          #locresl#[3] _ pvs1;
          #locresl#[4] _ pvs2;
          #rtnlist# _ LIST( MOVE #locresl#);
       END;
    = 67 %- filereturn -%:
       BEGIN
       %get the rjumpindex-th entry in the file return ring%
          filestraddr _ readfrring(da.dalink, rjumpindex : &srr);
          dismes (2,filestraddr);
       % save the values in the return list %
          #locresl#[1] _ readsrring(&srr, 0 : cc, pvs1, pvs2);
          #locresl#[2] _ cc;
          #locresl#[3] _ pvs1;
          #locresl#[4] _ pvs2;
          #locresl#[5] _ filestraddr;
          #locresl#[6] _ &srr;
          #rtnlist# _ LIST( MOVE #locresl#);
       END;
    ENDCASE
    IF NOT donthelp THEN
       BEGIN
       retry _ HELP (badarg, $"The type argument was incorrect.
       The allowed values are Filereturn or Statement.", cindex);
       REPEAT CASE (retry);
       END
    ELSE ABORT (badarg, $"The type argument was incorrect.");
 RETURN;
 END.

(xjmpcntdisp) PROCEDURE( % JUMP utility function which displays the
current contents of the content buffer (conreg) in the name window %
                                                                  8C
 %FORMALS%
    searchtype REF,
    rtnlist REF); %return list%
 LOCAL LIST locresl[2];
 LOCAL STRING locstr[100];
 LOCAL CONSTANT
    ctext = 1,
    cword = 3;
 % ------------- %
    CASE ELEM #searchtype# [cwtype] OF
```

```
            = 53:  %content%
               #locresl# [1] _ USE makedesc(uindex, ctext, FALSE);
            = 3:  %name%
               #locresl# [1] _ USE makedesc(uindex, cword, FALSE);
            ENDCASE ABORT(badarg, $"bad argument passed to
            xjmpcntdisp");
         locstr.L _ 0;
         CASE conreg.L OF
            = 0 : NULL;
            > 96 :  %truncate string%
               *locstr* _ *locstr*, *conreg* [1 TO 96], "";
            ENDCASE *locstr* _ *locstr*, *conreg*, "";
         #locresl# [2] _ USE makedesc(ustring, $conreg, FALSE);
         dismes(2, $locstr);
         #rtnlist# _ LIST( COPY #locresl#);
         RETURN ;
         END.

% Move edge code %
   = 21 %- edge -%:                          %frontend?%
      BEGIN
      &sourceda _ dsparea(boundry(source[1] : source[1], type));
         %get boundary nearest cursor%
      IF sourceda.dafrozen THEN
         ABORT(bademove,$"Can't move boundary of a frozen window");
      &destda _ destination;
      IF destda.dafrozen THEN
         ABORT(bademove,$"Can't move boundary of a frozen window");
      IF ELEM #dtype#[cwtype] = 207 %+ center +% THEN
         BEGIN
         &da _ findda(destination[1]);
         CASE type OF
            = lbound, = rbound:
               destination[1].xcord _ (da.daright-da.daleft)/2;
            = tbound, = bbound:
               destination[1].ycord _ (da.dabottom-da.datop)/2;
            ENDCASE;
         END;
      x _ destination[1].xcord - destda.daleft;
      IF destination[1].xcord < (destda.daright - destda.dahinc/2) THEN
         BEGIN
         DIV x / destda.dahinc, x, r;
         IF r >= (destda.dahinc/2) THEN BUMP x;
         x _ x * destda.dahinc;
         destination[1].xcord _ x + destda.daleft;
         END;
      y _ destination[1].ycord - destda.datop;
      IF destination[1].ycord < (destda.dabottom - destda.davinc/2)
      THEN
         BEGIN
         DIV y / destda.davinc, y, r;
         IF r >= (destda.davinc/2) THEN BUMP y;
         y _ y * destda.davinc;
         destination[1].ycord _ y + destda.datop;
         END;
      clearda(0); %erase entire text area%
```

```
      clrall(0, TRUE); %erase all line seg ref tables%
      movbndry(&sourceda, source[1], destination[1], type);
      IF NOT nodisplay THEN dpset(dspallf, endfil, endfil, endfil);
      RETURN;
      END;
% Insert edge code %
   = 33 %- edge -%: %of window%          %frontend?%
      BEGIN
      &da _ dsparea(ELEM #destptr#[wndw]);
      IF da.dafrozen THEN
         ABORT(badsplit,$"Cannot split a frozen window");
      boundry(destination[1] : cords, type);
         %ignore da returned%
      clearda(&da); %erase display image from da%
      clrall(&da, TRUE); %deallocate any strings%
      IF edgeloc = 54 %+ center +%  THEN
         BEGIN
         CASE type OF
            = tbound, = bbound:
               BEGIN
               x _ (da.daright-da.daleft)/2;
               cords.xcord _ (x/da.dahinc)*da.dahinc + da.daleft;
               END;
            = lbound, = rbound:
               BEGIN
               y _ (da.dabottom-da.datop)/2;
               cords.ycord _ (y/da.davinc)*da.davinc + da.datop;
               END;
            ENDCASE;
         END
      ELSE
         BEGIN
         cords.xcord _ ((cords.xcord)/da.dahinc)*da.dahinc;
         cords.ycord _ ((cords.ycord)/da.davinc)*da.davinc;
         END;
      CASE type OF
         = lbound, = rbound:
            IF NOT hsplit(&da, cords, lccords()) THEN
               ABORT(smallda,$"Display area too small");
         = tbound, = bbound:
            IF NOT vsplit(&da, cords, lccords()) THEN
               ABORT(smallda,$"Display area too small");
         ENDCASE;
      IF NOT nodisplay THEN dpset(dspallf, endfil, endfil, endfil);
      RETURN;
      END;
% Delete edge code %
   = 33 %- edge -%: %of window%          %frontend?%
      BEGIN
      &da _ dsparea(boundry(destination[1] : destination[1], type));
      cords _ lccords();
      IF NOT nodisplay THEN dpset(dspallf, endfil, endfil, endfil);
      clearda(0);
      clrall(0, TRUE);
      INVOKE (allrefresh);
      CASE type OF
```

```
    = lbound: %left edge of window DA to be deleted%
      BEGIN
      %da points to right window%
      IF da.daleft = taleft THEN
         ABORT(badedel,$"cannot delete margin edge");
      IF NOT da.dalneighbor THEN
         ABORT(noleftn,$"This window does not have a left
         neighbor");
      &deleteda _ dsparea(da.dalneighbor); %left window%
      IF cords.xcord > da.daleft THEN %keep windows to right of
      boundary%
         BEGIN
         fixbnd(TRUE, da.daleft, deleteda.daleft, TRUE);
         END
      ELSE %keep windows to left of boundary%
         BEGIN
         &deleteda _ &da := &deleteda;
         %now da is left and deleteda is right window%
         fixbnd(TRUE, da.daright, deleteda.daright, TRUE);
         END;
      END;
    = rbound: %right edge of window to be deleted%
      BEGIN
      IF da.daright = taright THEN
         ABORT(badedel,$"cannot delete margin edge");
      IF NOT da.darneighbor THEN
         ABORT(nortn,$"This window does not have a right
         neighbor");
      &deleteda _ dsparea(da.darneighbor); %right window%
      IF cords.xcord <= da.daright THEN %keep one to left of
      boundary%
         BEGIN
         fixbnd(TRUE, da.daright, deleteda.daright, TRUE);
         END
      ELSE %keep one to right of boundary%
         BEGIN
         &deleteda _ &da := &deleteda;
         fixbnd(TRUE, da.daleft, deleteda.daleft, TRUE);
         END;
      END;
    = tbound: %top edge of window to be deleted%
      BEGIN
      IF da.datop = tatop THEN
         ABORT(badedel,$"cannot delete margin edge");
      IF NOT da.datneighbor THEN
         ABORT(notopn,$"This window does not have a top
         neighbor");
      &deleteda _ dsparea(da.datneighbor); %top window%
      IF cords.ycord > da.datop THEN %keep window on bottom of
      edge%
         BEGIN
         fixbnd(FALSE, da.datop, deleteda.datop, TRUE);
         END
      ELSE %keep window on top of edge%
         BEGIN
         &deleteda _ &da := &deleteda;
```

```
            fixbnd(FALSE, da.dabottom, deleteda.dabottom, TRUE);
            END;
         END;
      = bbound: %bottom edge of window to be deleted%
         BEGIN
         IF da.dabottom = tabottom THEN
            ABORT(badedel,$"cannot delete margin edge");
         IF NOT da.dabneighbor THEN
            ABORT(nobtmn,$"This window does not have a bottom
            neighbor");
         &deleteda _ dsparea(da.dabneighbor); %bottom window%
         IF cords.ycord <= da.dabottom THEN %keep window on top of
         edge%
            BEGIN
            fixbnd(FALSE, da.dabottom, deleteda.dabottom, TRUE);
            END
         ELSE %keep window on bottom of edge%
            BEGIN
            &deleteda _ &da := &deleteda;
            fixbnd(FALSE, da.datop, deleteda.datop, TRUE);
            END;
         END;
      ENDCASE;
   DROP (allrefresh);
   fixbuf();
   END;
% Merge code -- never has worked %
   %merge PCP interface routine%
      (xmerge) %Execute Merge Command%                                  12A1
         PROCEDURE (
            %FORMALS%
               sourceptr REF,   %source pointer%
               destptr REF %destination pointer%
               );
         LOCAL retry REF, source REF, destination REF;
         %----------------------%
         IF &sourceptr THEN &source _ ELEM #sourceptr#[tppair];
         IF &destptr THEN &destination _ ELEM #destptr#[tppair];
         CASE ELEM #destptr#[cwtype] OF
            = 27 %- group -%, = 28 %- plex -%:
               BEGIN
               cmergro(&destination, &destination+d2sel, &source,
               &source+d2sel, ELEM #destptr#[wndw]);
               curmkr _ gethed(destination);  curmkr[1] _ 1;
               IF NOT nodisplay THEN dpset(dspstrc, destination,
               endfil, endfil);
               END;
            = 26 %- branch -%:
               BEGIN
               IF (destination := getsub(destination)) = destination OR
               (source := getsub(source)) = source THEN
                  ABORT (badmerge, $"Illegal merge.");
               destination[d2sel] _ getail(destination);
               source[d2sel] _ getail(source);
               REPEAT CASE(27 %+ group +%);
               END;
```

```
            ENDCASE
            IF NOT donthelp THEN
               BEGIN
               &retry _ HELP (badarg, $"The selection type was
               incorrect.  The allowed values are Branch, Group, and
               Plex.", cindex);
               destination _ ELEM #retry#[tppair];
               REPEAT CASE (ELEM #retry#[cwtype]);
               END
            ELSE ABORT (badarg, $"The selection type was incorrect.");
         RETURN;
         END.

   %merge core routines%
      (cmerge)          %Core NLS Merge Primitive%                    12B1
         PROCEDURE (outlst,inlst,kpr,percnt,copyf,window);
         merge (outlst,inlst,kpr,percnt,copyf,window);
         RETURN;
         END.

      (cmergro)         %Core NLS Merge Group Command%                12B2
         PROCEDURE (intobug1,intobug2,frombug1,trombug2,window);
         REF intobug1,intobug2,frombug1,frombug2;
         merggrp (&intobug1,&intobug2,&frombug1,&frombug2,window);
         RETURN;
         END.

      (cupdate)         %Core NLS Update Primitive%                   12B3
         PROCEDURE (outlst,inlst,kpr,udpr,percnt,initf);
         update (outlst,inlst,kpr,udpr,percnt,initf);
         RETURN;
         END.

% old gpget %
   (gpget)    PROCEDURE   % does Goto Programs Get Rel  File command %
                                                                    13A
      (adstr,   % address of parsed link data structure %
      shomes,  % If TRUE, do dismes's; FALSE, only error mssages form
      loader get out. %
      erstr REF); %string to put loader output (undefined etc) or 0 for
      TTY window %
      % invoke the TENLDR saved in our address space to load a rel file
        the file is loaded into the user program buffer, and the tenldr
      expects three arguments set up %
      LOCAL savprcadr, saveprogaddr, savsymloc, jfn, errs, lc, loader,
      entry, ddtloc, oneflag, exttlag, pgmfield, try, stname, i, ptype,
      dtstr[40];
      LOCAL TEXT POINTER tps,tpe, tp1, tp2;
      LOCAL STRING errstr[200];
      LOCAL STRING recurse[200];
      LOCAL STRING dirname[39];
      LOCAL STRING filename[39];
      LOCAL STRING extname[39];
      LOCAL STRING tempstr[50];
      LOCAL STRING extension[10];
      LOCAL STRING str[200];
```

```
    REF adstr, loader;
    % initialize %
       oneflag _ TRUE;
    % get the file name specified by the parsed link data structure %
       CASE lnbfls( 0, &adstr, $str) OF
          = lhostn: NULL;
          ENDCASE err($"Remote File Manipulations Not Implemented
          Yet");
    % edit the name string to get just the file name %
       extflag _ FALSE; % set to TRUE by the FIND stmt if extension
       is given %
       FIND SF(*str*) ^tps ^tpe;
       IF FIND tpe > "< [">] / "<^F> / "<ESC>] ^tps THEN NULL
       ELSE IF NOT FIND tpe > LD THEN
          err($"Illegal Rel File Name");
       IF FIND tps > ["."] ^tpe _tpe THEN
          BEGIN
          IF FIND LD THEN extflag _ TRUE;
          END
       ELSE
          IF FIND tps > ["<^F> / "<ESC>] ^tpe THEN
             BEGIN
             IF FIND LD THEN extflag _ TRUE;
             END
          ELSE
             IF NOT FIND tps > $LD (EOL / ENDCHR) ^tpe THEN
                err($"Illegal Rel File Name");
    % save away current state values %
       savsymloc _ symloc;
    % get the jfn for the rel file %
       % all of the following rather obtuse code attempts to figure
       out what file is to be loaded and what type it is. It is not
       necessary for the user to specify the extension name, as this
       grungy code manages to find that out. If no extension is
       given, then extensions are attempted in the following order:
       REL, CA, SK, SG, SUBSYS, CGR, PROC-REP. If this sequence
       cannot be satisfied in the default directory for links (for
       bugged links if no directory is specified) or in the connected
       directory ( for typed in links if no directory is specified)
       then the default directory for programs is used and the same
       sequence is repeated %
       *extension* _ *relext*; try _ 0;
       CASE jfn _ lgetjfn(0,$str,$extension,1811,$errstr) OF
          =FALSE: % getjfn failed, try some recovery strategy %
             BEGIN
             WHILE NOT extflag DO
                BEGIN % try to figure out what extension user wants %
                CASE try _ try + 1 OF
                   =1: *extension* _ *caext*;
                   =2: *extension* _ *skext*;
                   =3: *extension* _ *sgext*;
                   =4: *extension* _ *subext*;
                   =5: *extension* _ *cgrext*;
                   =6: *extension* _ *procext*;
                   ENDCASE EXIT LOOP;
                REPEAT CASE;
```

```
            END;
         % maybe try default user program directory %
            IF (oneflag := FALSE) AND
               ((*str*[1] # '<) OR (adstr[ue+1] = adstr[us+1]))
               THEN
                  BEGIN
                  *str* _ '<, *subdfdir*, '>, tps SE(*str*);
                  *extension* _ "REL"; try _ 0;
                  REPEAT CASE;
                  END;
            err($errstr);
            END;
         ENDCASE;
   % get rid of any ^F or altmodes in filename %
      jfntostr( jfn, $dirname, 010000B6);
      jfntostr( jfn, $filename, 001000B6);
      jfntostr( jfn, $extname, 000100B6);
   % find out what type of program for future instituting %
      ptype _ CASE TRUE OF
         = (*extname* = *relext*):  0;
         = (*extname* = *caext*):   1;
         = (*extname* = *skext*):   2;
         = (*extname* = *sgext*):   3;
         = (*extname* = *procext*): 4;
         = (*extname* = *cgrext*):  5;
         = (*extname* = *subext*):  6;
         ENDCASE 0;
   % open the file %
      IF NOT SKIP !openf (jfn, 4400002B5) THEN
         BEGIN
         IF NOT SKIP !rljfn( jfn ) THEN NULL;
         err( $"Open File Failed" );
         END;
   %adjust the user program buffer size if necessary to fit the
   program being loaded%
      gpadjbsz(jfn);
   % invoke the tenldr %
      IF shomes THEN dismes(1, $"Loading User Program");
      &loader _ nlsloader;
      errs _ loader(jfn, upgffbuf+1, upgbend, 0, &erstr :lc);
      IF shomes THEN dismes(0);
   % the tenldr returns an error count and the next address in the
   buffer %
      IF NOT errs THEN % good return %
         BEGIN
         % mark the block in the user program buffer %
            IF upgskix >= $upgsksz THEN
               ABORT (upgerr, $"no more room in user program stack")
            ELSE
               % load was successful and there was room in the
               buffer and there is room in stack %
               BEGIN
               upgstk [upgskix _ upgskix + 1] _ upgffbuf;
               % set string into user program buffer %
                  *upgnms* _ *upgnms*, SP, *filename*;
               END;
```

```
% update ddt's alt I-1 to include program just loaded %
   ddtloc _ ddtsptr; % get ptr to alt i -1 %
   [ddtloc] _ symloc; % stuff away new symloc %
% mark the block in the symbol table  %
   ddmrk[ddmrkx] _ savsymloc.RH - 1;
   IF (ddmrkx _ ddmrkx + 1 ) > ddmrkm THEN
      BEGIN
      ddmrkx _ ddmrkx - 1;
      err($"Mark stack overflow");
      END;
% lookup file name and set entry instruction %
   stname _
      IF ptype = 5 %cgr% THEN
         mrkglookup( $filename : entry )
      ELSE
         ddtlookup( $filename, FALSE : entry );
   IF stname  THEN
      [upgffbuf] _ 25489 + entry %set up entry inst %
   ELSE
      CASE ptype OF
         =0, =1, =2, =3:     % REL, CA, SK, SG %
            IF shomes THEN dismes(2,$"WARNING -- no entry
            to program ");
         =4:                              % PROC-REP %
            BEGIN
            *tempstr* _ "WARNING -- No Procedure Named ",
            *filename*;
            IF shomes THEN dismes(2,$tempstr);
            END;
         =5:                              % CGR %
            BEGIN
            *tempstr* _ "WARNING -- No Subsystem Named ",
            *filename*;
            IF shomes THEN dismes(2,$tempstr);
            END;
         =6:                              % SUBSYS %
            ABORT(ermunknown, $"No dispatch table for
            subsystem");
         ENDCASE;
% set new limit on buffer space %
    saveprogaddr _ upgffbuf := lc;
% institute program if so indicated by extension %
   CASE ptype OF
      = 1: % content analyzer %
         BEGIN
         IF shomes THEN dismes(2,$"Instituting User Program
         as a Content Analyzer");
         [lda()].dacacode _ saveprogaddr;
         END;
      = 2: % sort key %
         BEGIN
         IF shomes THEN dismes(2,$"Instituting User Program
         as a Sort Key Program");
         [lda()].daukeycod _ saveprogaddr;
         END;
      = 3: % sequence generator %
```

```
                BEGIN
                IF shomes THEN dismes(2,$"Instituting User Program
                as a Sequence Generator");
                [lda()].dausqcod _ saveprogaddr;
                END;
            = 4: % procedure replace %
                IF stname THEN
                   BEGIN
                   IF rplproc( $filename, $filename) THEN
                      *tempstr* _ "Procedure ", *filename*, "
                      Replaced"
                   ELSE
                      *tempstr* _ "Old Procedure ", *filename*, "
                      Does Not Exist";
                   IF shomes THEN dismes(2, $tempstr);
                   END;
            = 5:  % cgr %
                IF stname THEN
                   BEGIN
                   upgstk[upgskix].LH _ entry;
                   % Noted in BEIGNORE
                   dfnsubsys( entry, gtctrlbits(entry), $nlssubs
                   );
                   dfnsubsys( entry, gtctrlbits(entry), $allsubs
                   );
                   %
                   *tempstr* _ "Subsystem ", *filename*, " Now
                   Available (Attached)";
                   IF shomes THEN dismes(2, $tempstr);
                   END;
            = 6:  % subsys %
                BEGIN
                *tempstr* _ *filename*, ".GRAM";
                %possible that a cll to a FE parse function to
                switch to a new grammar should go here%
                % define the package just loaded%
                   upgstk[upgskix].LH _ crtlpkg( $filename, entry,
                   $dirname, 0, 0, 0);
                END;
            ENDCASE;
        END
    ELSE
        BEGIN
        symloc _ savsymloc; % reset symloc as it may be clobbered
        by an error in loading %
        err( $"Error in Loading");
        END;
RETURN;

(nofront) CATCHPHRASE;                                          13A29
    BEGIN
    DISABLE(nofront);
    ABORT(err, $"Can't load frontend for this SUBSYS");
    END;

END.
```

```
% old xjump %
   (xjump) %Execute Jump Command%
   PROCEDURE (                                                          14A
      %FORMALS%
         entity REF,     %type of jump%
         destptr REF,    %dest record%
         vs REF,         %view specs%
         window,         %window number%
         retlist REF     %return list %
         );
      LOCAL retry, destination REF, da REF, start, oldda REF, fileno,
      linkp REF, vsupdate, adstr[40], destarea[20], vswrd1, vswrd2;
      LOCAL TEXT POINTER t1, t2, csp,
         srchtp1, srchtp2;  %keep these in order%
      LOCAL STRING locstr[200], badmsg[250];
      %----------------------%
      INVOKE (catjump);
      IF &destptr THEN &destination _ ELEM #destptr#[tppair]
      ELSE
         BEGIN  %use previous search pattern%
         FIND SF(*conreg*) ^srchtp1 SE(*conreg*) ^srchtp2;
         &destination _ $srchtp1;
         END;
      &da _ cspupdate _ dsparea(window);
      IF NOT &vs THEN
         BEGIN
         vswrd1 _ da.davspec;
         vswrd2 _ da.davspc2;
         END
      ELSE
         BEGIN
         vswrd1 _ vs;
         vswrd2 _ vs[1];
         END;
      csp _ da.dacsp;  csp[1] _ da.daccnt;
      cspvs _ da.davspec;
      cspvs[1] _ da.davspc2;
      vsupdate _ TRUE;
      start _ &destination;
      CASE ELEM #entity#[cwtype] OF
         = 29 %- statement -%:
            BEGIN
            (xj0):                                                      14A17A2
            IF cspupdate THEN
               BEGIN
               curmkr _ destination;
               curmkr[1] _ IF nlmode = fulldisplay THEN 1 ELSE
               destination[1];
               END;
            IF vsupdate THEN
               BEGIN
               cspvs _ vswrd1;
               cspvs[1] _ vswrd1[1];
               END;
            IF NOT nodisplay THEN dpset(dspjpf, curmkr, endfil,
```

```
    endfil);
    DROP (catjump);
    RETURN;
    END;
= 52 %- successor -%:  *locstr* _ ".s";
= 53 %- predecessor -%:        *locstr* _ ".p";
= 54 %- up -%:          *locstr* _ ".u";
= 55 %- down -%:                *locstr* _ ".d";
= 56 %- head -%:                *locstr* _ ".h";
= 57 %- tail -%:                *locstr* _ ".t";
= 58 %- end -%:                 *locstr* _ ".e";
= 59 %- back -%:                *locstr* _ ".b";
= 60 %- origin -%:              *locstr* _ ".o";
= 61 %- next -%:                *locstr* _ ".n";
% jump FILE RETURN  and jump RETURN replaced by xjumpreturn -
code commented out%
= 30 %- link -%:
    BEGIN
    vsupdate _ FALSE;
    IF destination.stastr THEN
       BEGIN
       start _ $csp;
       t1 _ destination; t1[1] _ destination[1];
       t2 _ [&destination+d2sel];
          t2[1] _ [&destination+d2sel+1];
       END
    ELSE
       BEGIN
       lnkprs( &destination, $adstr);
       t1 _ adstr[ls]; t1[1] _ adstr[ls+1];
       t2 _ adstr[le]; t2[1] _ adstr[le+1];
       END;
    GOTO xj2;
    END;
= 32 %- name -%:
    BEGIN
    t1 _ destination;
    t1[1] _ [&destination+d2sel+1];
    *locstr* _ destination t1;
    start _ $csp;
    END;
= 63 %- filenamed -%:
    BEGIN
    t1 _ destination;
    t1[1] _ [&destination+d2sel+1];
    *locstr* _ "(, destination t1, ",)";
    start _ $csp;
    END;
= 51 %- file -%:
    BEGIN
    lnkprs(&destination, $adstr);
    t1 _ adstr[ls];  t1[1] _ adstr[ls+1];
    t2 _ adstr[fe];  t2[1] _ adstr[fe+1];
    *locstr* _ t1 t2, ",)";
    start _ $csp;
    END;
```

```
        = 64 %- firstname -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ ".o*", destination t1;
            start _ $csp;
            END;
        = 65 %- nextname -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ "*", destination t1;
            start _ $csp;
            END;
        = 66 %- extname -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ "$, destination t1;
            start _ $csp;
            END;
        = 68 %- firstcontent -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ ".o", "", destination t1, "", "=C";
            start _ $csp;
            END;
        = 69 %- nextcontent -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ ".n", "", destination t1, "", "=C";
            start _ $csp;
            END;
        = 70 %- firstword -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ ".o", "", destination t1, "", "=W";
            start _ $csp;
            END;
        = 71 %- nextword -%:
            BEGIN
            t1 _ destination;
            t1[1] _ [&destination+d2sel+1];
            *locstr* _ ".n", "", destination t1, "", "=W";
            start _ $csp;
            END;
        ENDCASE
        IF NOT donthelp THEN
            BEGIN
            *badmsg* _ "The type argument was incorrect.  The allowed
            values are Back, Down, End, Extname, File, Filenamed,
            Filereturn, Firstcontent, ";
            *badmsg* _ *badmsg*, "Firstname, Firstword, Head, Link,
```

```
        Name, ";
        *badmsg* _ *badmsg*,  "Next, Nextcontent, Nextname,
        Nextword, Origin, Predecessor, Return, Statement,
        Successor, Tail, or Up.";
        retry _ HELP (badarg, $badmsg, cindex);
        REPEAT CASE (retry);
        END
     ELSE ABORT (badarg, $"The type argument was incorrect.");
  (xj1): %set up text pointers and evaluate address expression%
                                                              14A18
  FIND SF(*locstr*) ^t1 SE(*locstr*) ^t2;
  (xj2): %assuming the text pointers are set up, evaluate address
  expression%                                                 14A20
  IF vsupdate THEN
     BEGIN
     da.davspec _ vswrd1;
     da.davspc2 _ vswrd1[1];
     END;
  vswrd1 _ caddexp($t1, $t2, &da, start : vswrd1[1], cspcacode,
  cspusqcod%, usesrr%);
  usesrr _ fakesrr;
  IF vsupdate THEN
     BEGIN
     da.davspec _ cspvs;
     da.davspc2 _ cspvs[1];
     END;
  vsupdate _ TRUE;
  destination _ destination[d2sel] _ [start];
     destination[1] _ destination[d2sel+1] _ [start+1];
  GOTO xj0;
  RETURN;
  %Catchphrases%                                               14A29
     (catjump) CATCHPHRASE();                                 14A29A
        BEGIN
        CASE SIGNALTYPE OF
           = aborttype :  %stop cmdfinish from reformatting
           display%
              cspupdate _ cspcacode _ cspusqcod _ 0;
           ENDCASE;
        CONTINUE;
        END;
  END.
```