(aarray)	<nine, library, 06322>	EXT	1B20
(always)	<nine, library, 06364>	EXT STRING	1B2AA3
(andtable)	<nine, library, 06333>	EXT	1B2A@2
(anyaction)	<nine, library, 06489>	EXT	1B2AD2
(argxcw)	<nine, library, 010354>	EXT CONSTANT =1	1B2AE3
(argxvs)	<nine, library, 010362>	EXT CONSTANT =2	1B2AE4
(bvar1)	<nine, library, 06401>	EXT STRING	1B2AC4
(bvar10)	<nine, library, 06410>	EXT STRING	1B2AC1
(bvar11)	<nine, library, 06411>	EXT STRING	1B2AC1
(bvar12)	<nine, library, 06412>	EXT STRING	1B2AC1
(bvar13)	<nine, library, 06413>	EXT STRING	1B2AC1
(bvar14)	<nine, library, 06414>	EXT STRING	1B2AC1
(bvar15)	<nine, library, 06415>	EXT STRING	1B2AC1
(bvar16)	<nine, library, 06416>	EXT STRING	1B2AC1
(bvar17)	<nine, library, 06417>	EXT STRING	1B2AC2
(bvar18)	<nine, library, 06418>	EXT STRING	1B2AC2
(bvar19)	<nine, library, 06419>	EXT STRING	1B2AC2
(bvar2)	<nine, library, 06402>	EXT STRING	1B2AC5
(bvar20)	<nine, library, 06420>	EXT STRING	1B2AC2
(bvar21)	<nine, library, 06421>	EXT STRING	1B2AC2
(bvar22)	<nine, library, 06422>	EXT STRING	1B2AC2
(bvar23)	<nine, library, 06423>	EXT STRING	1B2AC2
(bvar24)	<nine, library, 06424>	EXT STRING	1B2AC2
(bvar25)	<nine, library, 06425>	EXT STRING	1B2AC2
(bvar26)	<nine, library, 06426>	EXT STRING	1B2AC2
(bvar27)	<nine, library, 06427>	EXT STRING	1B2AC3
(bvar28)	<nine, library, 06428>	EXT STRING	1B2AC3
(bvar29)	<nine, library, 06429>	EXT STRING	1B2AC3
(bvar3)	<nine, library, 06403>	EXT STRING	1B2AC6
(bvar30)	<nine, library, 06430>	EXT STRING	1B2AC3
(bvar31)	<nine, library, 06431>	EXT STRING	1B2AC3
(bvar32)	<nine, library, 06432>	EXT STRING	1B2AC3
(bvar33)	<nine, library, 06433>	EXT STRING	1B2AC3
(bvar34)	<nine, library, 06434>	EXT STRING	1B2AC3
(bvar35)	<nine, library, 06435>	EXT STRING	1B2AC3
(bvar36)	<nine, library, 06436>	EXT STRING	1B2AC3
(bvar37)	<nine, library, 06437>	EXT STRING	1B2AC4
(bvar38)	<nine, library, 06438>	EXT STRING	1B2AC4
(bvar39)	<nine, library, 06439>	EXT STRING	1B2AC4
(bvar4)	<nine, library, 06404>	EXT STRING	1B2AC7
(bvar40)	<nine, library, 06440>	EXT STRING	1B2AC4
(bvar5)	<nine, library, 06405>	EXT STRING	1B2AC8
(bvar6)	<nine, library, 06406>	EXT STRING	1B2AC9
(bvar7)	<nine, library, 06407>	EXT STRING	1B2AC1
(bvar8)	<nine, library, 06408>	EXT STRING	1B2AC1
(bvar9)	<nine, library, 06409>	EXT STRING	1B2AC1
(bvarstr)	<nine, library, 06442>	EXT	1B2AC4
(carray)	<nine, library, 06323>	EXT	1B2P
(cfstid)	<nine, library, 06306>	EXT	1B2H5
(cisfile)	<nine, library, 07465>	PROCEDURE	1C5
(cispjtn)	<nine, library, 07476>	PROCEDURE	1C6
(cmdbra)	<nine, library, 06395>	EXT STRING	1B2AB5
(cmperr)	<nine, library, 06523>	EXT	1B2AD2
(cmptile)	<nine, library, 07543>	PROCEDURE	1C7
(cmpok)	<nine, library, 06524>	EXT	1B2AD2
(cmpptr)	<nine, library, 09889>	EXT TEXT POINTER	1B2Q

(cnvvsp)	<nine, library, 010464>	PROCEDURE	1C43A
(compiler)	<nine, library, 010402>	EXT STRING	1B2Z
(condcnt)	<nine, library, 06295>	EXT	1B2P
(constid)	<nine, library, 06313>	EXT	1B2H12
(cplex)	<nine, library, 07649>	PROCEDURE	1C8
(cpysplex)	<nine, library, 07768>	PROCEDURE	1C10
(cqplex)	<nine, library, 07747>	PROCEDURE	1C9
(createdate)	<nine, library, 06389>	EXT STRING	1B2AA5
(crehis)	<nine, library, 07785>	PROCEDURE	1C11
(csabrt)	<nine, library, 06359>	EXT STRING	1B2AA2
(cscomm)	<nine, library, 06351>	EXT STRING	1B2AA1
(cscomp)	<nine, library, 06338>	EXT STRING	1B2AA3
(cscond)	<nine, library, 06344>	EXT STRING	1B2AA9
(cscopyprinter)	<nine, library, 010335>	EXT STRING	1B2AA2
(csdate)	<nine, library, 06355>	EXT STRING	1B2AA2
(csdcml)	<nine, library, 06354>	EXT STRING	1B2AA1
(csever)	<nine, library, 06336>	EXT STRING	1B2AA1
(csignr)	<nine, library, 06358>	EXT STRING	1B2AA2
(csincl)	<nine, library, 06345>	EXT STRING	1B2AA1
(csindx)	<nine, library, 06342>	EXT STRING	1B2AA7
(cslist)	<nine, library, 06337>	EXT STRING	1B2AA2
(cslogo)	<nine, library, 06360>	EXT STRING	1B2AA2
(csocti)	<nine, library, 06353>	EXT STRING	1B2AA1
(csoutp)	<nine, library, 06356>	EXT STRING	1B2AA2
(cspril)	<nine, library, 06339>	EXT STRING	1B2AA4
(cspind)	<nine, library, 06348>	EXT STRING	1B2AA1
(csplib)	<nine, library, 06347>	EXT STRING	1B2AA1
(cspsys)	<nine, library, 06349>	EXT STRING	1B2AA1
(csptyp)	<nine, library, 06346>	EXT STRING	1B2AA1
(csrept)	<nine, library, 06357>	EXT STRING	1B2AA2
(csrunt)	<nine, library, 06341>	EXT STRING	1B2AA6
(csstat)	<nine, library, 06352>	EXT STRING	1B2AA1
(cssysg)	<nine, library, 06340>	EXT STRING	1B2AA5
(csupdt)	<nine, library, 06343>	EXT STRING	1B2AA8
(dates)	<nine, library, 06530>	EXT	1B2AD4
(default)	<nine, library, 06363>	EXT STRING	1B2AA2
(dindex)	<nine, library, 06398>	EXT	1B2AC1
(dofile)	<nine, library, 07804>	PROCEDURE	1C12
(dsply)	<nine, library, 010356>	EXT CONSTANT =51	1B2AE2
(edrive)	<nine, library, 06553>	EXT	1B2AD2
(fadd)	<nine, library, 06369>	EXT STRING	1B2AA3
(fand)	<nine, library, 06367>	EXT STRING	1B2AA3
(fdrcomment)	<nine, library, 010456>	EXT STRING	1C3A3
(fdrparams)	<nine, library, 010457>	EXT STRING	1C3A4
(fdrstmnt)	<nine, library, 010454>	EXT STRING	1C3A1
(fdrstnum)	<nine, library, 010455>	EXT STRING	1C3A2
(fdrtype)	<nine, library, 010458>	EXT STRING	1C3A5
(fdrwidth)	<nine, library, 010459>	EXT STRING	1C3A6
(fget)	<nine, library, 06366>	EXT STRING	1B2AA3
(filcmp)	<nine, library, 06527>	EXT	1B2AD2
(filenum)	<nine, library, 06297>	EXT	1B2D
(filepc)	<nine, library, 06519>	EXT	1B2AD2
(filncmp)	<nine, library, 06528>	EXT	1B2AD2
(finder)	<nine, library, 09509>	PROCEDURE	1C42
(fname)	<nine, library, 06330>	EXT STRING	1B2W
(fndqual)	<nine, library, 07957>	PROCEDURE	1C13

(fnot)	<nine, library, 06374>	EXT STRING	1B2AA4
(for)	<nine, library, 06365>	EXT STRING	1B2AA3
(fsetf)	<nine, library, 06372>	EXT STRING	1B2AA3
(fsetm)	<nine, library, 06373>	EXT STRING	1B2AA3
(fsett)	<nine, library, 06371>	EXT STRING	1B2AA3
(fsub)	<nine, library, 06370>	EXT STRING	1B2AA3
(fsuper)	<nine, library, 08031>	PROCEDURE	1C14
(fxor)	<nine, library, 06368>	EXT STRING	1B2AA3
(gabort)	<nine, library, 06531>	EXT	1B2AD5
(garray)	<nine, library, 06321>	EXT	1B2N
(gcompile)	<nine, library, 06549>	EXT	1B2AD2
(gconcnt)	<nine, library, 06539>	EXT	1B2AD1
(gcopyprinter)	<nine, library, 010339>	EXT	1B2AD1
(getcompile)	<nine, library, 08077>	PROCEDURE	1C15
(gettnum)	<nine, library, 08104>	PROCEDURE	1C16
(getwrtdate)	<nine, library, 08124>	PROCEDURE	1C17
(gi)	<nine, library, 06325>	EXT	1B2S
(gignore)	<nine, library, 06535>	EXT	1B2AD9
(ginccnt)	<nine, library, 06540>	EXT	1B2AD1
(glogout)	<nine, library, 06551>	EXT	1B2AD2
(gmstid)	<nine, library, 08136>	PROCEDURE	1C18
(goutput)	<nine, library, 06537>	EXT	1B2AD1
(gpindex)	<nine, library, 06544>	EXT	1B2AD1
(gplibbranch)	<nine, library, 06543>	EXT	1B2AD1
(gprint)	<nine, library, 06541>	EXT	1B2AD1
(gpsysguide)	<nine, library, 06545>	EXT	1B2AD2
(gptypescript)	<nine, library, 06542>	EXT	1B2AD1
(grepeat)	<nine, library, 06532>	EXT	1B2AD6
(grunfile)	<nine, library, 06534>	EXT	1B2AD8
(gsupersysgd)	<nine, library, 06550>	EXT	1B2AD2
(gsysgd)	<nine, library, 06538>	EXT	1B2AD1
(gtdate)	<nine, library, 06921>	PROCEDURE	1B3A
(guindex)	<nine, library, 08186>	PROCEDURE	1C19
(gupdate)	<nine, library, 06548>	EXT	1B2AD2
(guptype)	<nine, library, 06546>	EXT	1B2AD2
(incfile)	<nine, library, 09872>	EXT STRING	1B2X
(incidcnt)	<nine, library, 06296>	EXT	1B2C
(incstid)	<nine, library, 06314>	EXT	1B2H13
(infile)	<nine, library, 06298>	EXT	1B2E
(intnok)	<nine, library, 06515>	EXT	1B2AD2
(intnrun)	<nine, library, 06517>	EXT	1B2AD2
(intok)	<nine, library, 06516>	EXT	1B2AD2
(intrun)	<nine, library, 06518>	EXT	1B2AD2
(initprnt)	<nine, library, 08228>	PROCEDURE	1C20
(inproc)	<nine, library, 06300>	EXT	1B2G
(inrecord)	<nine, library, 06299>	EXT	1B2F
(inxdate)	<nine, library, 06502>	EXT	1B2AD2
(isysgd)	<nine, library, 08264>	PROCEDURE	1C21
(icdate)	<nine, library, 06498>	EXT	1B2AD2
(ihsstid)	<nine, library, 06305>	EXT	1B2H4
(libdate)	<nine, library, 06499>	EXT	1B2AD2
(libfile)	<nine, library, 06391>	EXT STRING	1B2AB1
(libhis)	<nine, library, 06396>	EXT STRING	1B2AB6
(library)	<nine, library, 09840>	EXT	1B2A1
(libsrttime)	<nine, library, 06316>	EXT	1B2I
(lidate)	<nine, library, 06496>	EXT	1B2AD2

(sdrive)	<nine, library, 06533>	EXT	1B2AD7
(sourcedate)	<nine, library, 06500>	EXT	1B2AD2
(sprstid)	<nine, library, 06303>	EXT	1B2H2
(srcfil)	<nine, library, 06392>	EXT STRING	1B2AB2
(srcstid)	<nine, library, 06307>	EXT	1B2H6
(srcwrd)	<nine, library, 06317>	EXT	1B2J
(srtjfns)	<nine, library, 06320>	EXT	1B2M
(supfile)	<nine, library, 09873>	EXT STRING	1B2Y
(sysorg)	<nine, library, 06302>	EXT	1B2H1
(sysstid)	<nine, library, 06311>	EXT	1B2H10
(tfe)	<nine, library, 06381>	EXT STRING	1B2AA4
(tff)	<nine, library, 06376>	EXT STRING	1B2AA4
(tfg)	<nine, library, 06383>	EXT STRING	1B2AA4
(tfi)	<nine, library, 06385>	EXT STRING	1B2AA5
(tfm)	<nine, library, 06377>	EXT STRING	1B2AA4
(tfne)	<nine, library, 06382>	EXT STRING	1B2AA4
(tfnr)	<nine, library, 06379>	EXT STRING	1B2AA4
(tfng)	<nine, library, 06384>	EXT STRING	1B2AA5
(tfni)	<nine, library, 06386>	EXT STRING	1B2AA5
(tfnm)	<nine, library, 06380>	EXT STRING	1B2AA4
(tfnt)	<nine, library, 06378>	EXT STRING	1B2AA4
(tft)	<nine, library, 06375>	EXT STRING	1B2AA4
(topalmost)	<nine, library, 07054>	PROCEDURE	1C4
(topstid)	<nine, library, 06304>	EXT	1B2H3
(totalerrors)	<nine, library, 06521>	EXT	1B2AD2
(typwrtr)	<nine, library, 010357>	EXT CONSTANT =52	1B2AE1
(uflag)	<nine, library, 06400>	EXT STRING	1B2AC3
(upconcnt)	<nine, library, 09363>	PROCEDURE	1C36
(upfile)	<nine, library, 09377>	PROCEDURE	1C37
(uphstid)	<nine, library, 06309>	EXT	1B2H8
(upinccnt)	<nine, library, 09418>	PROCEDURE	1C38
(upsysgd)	<nine, library, 09432>	PROCEDURE	1C39
(usertlags)	<nine, library, 06529>	EXT	1B2AD3
(writedate)	<nine, library, 06387>	EXT STRING	1B2AA5
(wrtmsg)	<nine, library, 09453>	PROCEDURE	1C40
(xlbevtop)	<nine, library, 06818>	PROCEDURE	1B3D
(xlchkadr)	<nine, library, 010577>	PROCEDURE	1B3E
(xldelhis)	<nine, library, 06864>	PROCEDURE	1B3F
(xlicompile)	<nine, library, 06990>	PROCEDURE	1B3G
(xliinit)	<nine, library, 07031>	PROCEDURE	1B3H
(xlreset)	<nine, library, 010494>	PROCEDURE	1B3I
(xortable)	<nine, library, 06334>	EXT	1B2A@3

```
< NINE, LIBRARY.NLS;28, >, 27-May-78 21:14 BLP ;;;;
%source-code%                                                          1
   %grammar% FILE library  % (arcsubsys, cml10,)  (relnine,
   library.cgr,) %                                                     1A
      % COMPILE INSTRUCTIONS %
        INCLUDE <nine, nls-grammar, flags !subsystems>
      % DECLARATIONS %
        INCLUDE <nine, nls-grammar, declarations !universal>
        DECLARE COMMAND WORD
           "DISPLAY" = 51,
           "TYPEWRITER" = 52;
        DECLARE FUNCTION
           xlbevtop, xlchkadr, xldelhis, xlicompile, xliinit, xlreset,
           xsimulate;
        DECLARE VARIABLE namfil, diagnos;
      % COMMON RULES %
        INCLUDE <nine, nls-grammar, rules !universal>
      % COMMANDS % SUBSYSTEM library KEYWORD "LIBRARY"
        INITIALIZATION ziinit = xliinit();
        COMMAND  zpcompile = "COMPILE"    sent _ "FILE"
           <"at"> source _ DSEL(#"STATEMENT")
           <"using"> param _ LSEL(#"OLDFILENAME")
           <"to file"> namfil _ LSEL(#"NEWFILENAME")
           ( "RECORD" <"messages at"> diagnos _ DSEL(#"STATEMENT")
              CONFIRM
              xlicompile( sent, source, param, namfil, diagnos )
            / CONFIRM
              xlicompile( sent, source, param, namfil, NULL )
           );
        %COMMAND ziinclude = "INSTITUTE" <"include sequence
        generator">
           CONFIRM xliinstitute();%
        COMMAND zdelete="DELETE"
           <"history before date and time"> param _ LSEL(#"TEXT")
           <"in library form at"> source _ DSEL(#"BRANCH")
           CONFIRM
           xldelhis( param, source);
        COMMAND zprocess="PROCESS"
           <"library form at"> source _ DSEL(#"BRANCH")
           CONFIRM
           xlchkadr(source)
              %check for valid address before simulating a typewriter.
               Otherwise if a failure occurs during argument
              conversion, the command will be aborted by the CLI and
              the terminal will be left in typewriter mode.%
           ( IF DISPLAY  %simulate typewriter mode%
              xsimulate( #"TYPEWRITER" )
              simtty()
              ttysim _ TRUE
           / IF NOT DISPLAY
              ttysim _ FALSE )
           xlbevtop( source)  %do the Process command%
           ( IF ttysim  %go back to display mode%
              simdisplay()
              xlreset(#"DISPLAY", source)  %reset display%
              ttysim _ FALSE
```

```
      / IF NOT ttysim );
    INCLUDE <nine, nls-grammar, commands !universal>
    END.
FINISH
```

```
%be% FILE librarybe % using (arcsubsys, 1109,) to (relnine,
library.subsys,) %                                                    1B
   ALLOW!
   % Declarations %
      % Dispatch Table %
         (library) EXTERNAL _ (                                       1B2A1
            $"XLBEVTOP", $xlbevtop,
            $"XLCHKADR", $xlchkadr,
            $"XLDELHIS", $xldelhis,
            $"XLICOMPILE", $xlicompile,
            $"XLIINIT", $xliinit,
            $"XLRESET", $xlreset,
            0,0);
      (condcnt) EXTERNAL;     % conditional count while printing %
                                                                     1B2B
      (incldcnt) EXTERNAL;    % include count while printing %     1B2C
      (filenum) EXTERNAL;     % filenumber while printing %        1B2D
      (infile) EXTERNAL;      % indicates within code file while
      printing %                                                     1B2E
      (inrecord) EXTERNAL;    % indicates within record declaration
      while printing %                                               1B2F
      (inproc) EXTERNAL;      % indicates within procedure body while
      printing %                                                     1B2G
      % stids %
         (sysorg) EXTERNAL;   % stid index file origin %           1B2H1
         (sprstid) EXTERNAL;  % stid sysguide file origin %        1B2H2
         (topstid) EXTERNAL;  % stid library branch %              1B2H3
         (lhsstid) EXTERNAL;  % stid library history branch %     1B2H4
         (cfstid) EXTERNAL;   % stid current library branch statement
         %                                                          1B2H5
         (srcstid) EXTERNAL;  % stid current source file origin %
                                                                    1B2H6
         (procstid) EXTERNAL;         % stid current process hstry
         stmnt in library branch %                                  1B2H7
         (uphstid) EXTERNAL;  % stid current update history stmnt in
         library branch %                                           1B2H8
         (prntstid) EXTERNAL;         % stid current print history
         stmnt in library branch %                                  1B2H9
         (sysstid) EXTERNAL;  % stid current index stmnt in library
         branch %                                                  1B2H10
         (relstid) EXTERNAL;  % stid current compile stmnt in library
         branch %                                                  1B2H11
         (constid) EXTERNAL;  % stid current conditional stmnt in
         library branch %                                          1B2H12
         (incstid) EXTERNAL;  % stid current include stmnt in library
         branch %                                                  1B2H13
      (libsrttime) EXTERNAL;  % library processing start time %    1B2I
      (srcwrd) EXTERNAL;      % source file write date; 35M if file
      has PC %                                                       1B2J
      (reldate) EXTERNAL;     % rel file date %                    1B2K
      (newsysgdcnt) EXTERNAL;         % count of new sysgds generated
      %                                                            1B2L
      (srtjfns) EXTERNAL;     % starting primary jfns %            1B2M
      (garray) EXTERNAL [40];                                      1B2N
      (aarray) EXTERNAL [40]; % parsed link array for compile at
```

```
branch address %                                                    1B20
(carray) EXTERNAL [40];  % parsed link array for compiler %
                                                                    1B2P
(cmpptr) EXTERNAL TEXT POINTER;  %text ptr. to compiler link%
                                                                    1B2Q
(relptr) EXTERNAL TEXT POINTER;  %text ptr. to rel file link%
                                                                    1B2R
(gi) EXTERNAL; % so we dont get auto-logged out %                   1B2S
(loaded) EXTERNAL = FALSE;     % so we only load once %           1B2T
(relname) EXTERNAL STRING [200]; % rel file name for compiles
%                                                                   1B2U
(qpfilename) EXTERNAL STRING [200];                                 1B2V
(fname) EXTERNAL STRING [70];                                       1B2W
(incfile) EXTERNAL STRING = "(include.sg,)";                        1B2X
   %include sequence generator file%
(supfile) EXTERNAL STRING = "(relnine, libsup.rel,)";               1B2Y
   %library support routines file%
(compiler) EXTERNAL STRING = "<arcsubsys, 1109, >";                 1B2Z
   %name of compiler for LIBRARIAN generated COMPILE
   statements%
% table combinations              NN  NM  NY  MN  MM  MY  YN
YM  YY %
   (ortable) EXTERNAL = ( -1,  0,  1,  0,  0,  1,  1,  1,  1);
                                                                    1B2A@1
   (andtable) EXTERNAL = ( -1, -1, -1, -1,  0,  0, -1,  0,
   1);                                                              1B2A@2
   (xortable) EXTERNAL = ( -1,  0,  1,  0,  0,  0,  1,  0,
   -1);                                                             1B2A@3
% qualifiers & command word strings %
   (csever) EXTERNAL STRING = "everything";                         1B2AA1
   (cslist) EXTERNAL STRING = "list";                               1B2AA2
   (cscomp) EXTERNAL STRING = "compile";                            1B2AA3
   (cspfil) EXTERNAL STRING = "print-file";                         1B2AA4
   (cssysg) EXTERNAL STRING = "sysguide";                           1B2AA5
   (csrunf) EXTERNAL STRING = "runfile";                            1B2AA6
   (csindx) EXTERNAL STRING = "index";                              1B2AA7
   (csupdt) EXTERNAL STRING = "update";                             1B2AA8
   (cscond) EXTERNAL STRING = "conditional";                        1B2AA9
   (csincl) EXTERNAL STRING = "include";                           1B2AA10
   (csptyp) EXTERNAL STRING = "print-typescript";                  1B2AA11
   (csplib) EXTERNAL STRING = "print-library-branch";              1B2AA12
   (cspind) EXTERNAL STRING = "print-index";                       1B2AA13
   (cspsys) EXTERNAL STRING = "print-sysguide";                    1B2AA14
   %(csdtch) EXTERNAL STRING = "detach"; %%not in NLS10%
   (cscomm) EXTERNAL STRING = "comment";                           1B2AA16
   (csstat) EXTERNAL STRING = "status";                            1B2AA17
   (csoctl) EXTERNAL STRING = "octal";                             1B2AA18
   (csdcml) EXTERNAL STRING = "decimal";                           1B2AA19
   (csdate) EXTERNAL STRING = "date";                              1B2AA20
   (csoutp) EXTERNAL STRING = "output";                            1B2AA21
   (csrept) EXTERNAL STRING = "repeat";                            1B2AA22
   (csignr) EXTERNAL STRING = "ignore";                            1B2AA23
   (csabrt) EXTERNAL STRING = "abort";                             1B2AA24
   (cslogo) EXTERNAL STRING = "logout";                            1B2AA25
   (cscopyprinter) EXTERNAL STRING = "copyprinter";                1B2AA26
   %(cswait) EXTERNAL STRING = "wait";%
```

```
    (never) EXTERNAL STRING = "never";                          1B2AA28
    (default) EXTERNAL STRING = "default";                      1B2AA29
    (always) EXTERNAL STRING = "always";                        1B2AA30
    (for) EXTERNAL STRING = "for";                              1B2AA31
    (fget) EXTERNAL STRING = "fget";                            1B2AA32
    (fand) EXTERNAL STRING = "fand";                            1B2AA33
    (fxor) EXTERNAL STRING = "fxor";                            1B2AA34
    (fadd) EXTERNAL STRING = "fadd";                            1B2AA35
    (fsub) EXTERNAL STRING = "fsub";                            1B2AA36
    (fsett) EXTERNAL STRING = "fsett";                          1B2AA37
    (fsetf) EXTERNAL STRING = "fsetf";                          1B2AA38
    (fsetm) EXTERNAL STRING = "fsetm";                          1B2AA39
    (fnot) EXTERNAL STRING = "fnot";                            1B2AA40
    (tft) EXTERNAL STRING = "tft";                              1B2AA41
    (tff) EXTERNAL STRING = "tff";                              1B2AA42
    (tfm) EXTERNAL STRING = "tfm";                              1B2AA43
    (tfnt) EXTERNAL STRING = "tfnt";                            1B2AA44
    (tfnf) EXTERNAL STRING = "tfnf";                            1B2AA45
    (tfnm) EXTERNAL STRING = "tfnm";                            1B2AA46
    (tfe) EXTERNAL STRING = "tfe";                              1B2AA47
    (tfne) EXTERNAL STRING = "tfne";                            1B2AA48
    (tfg) EXTERNAL STRING = "tfg";                              1B2AA49
    (tfng) EXTERNAL STRING = "tfng";                            1B2AA50
    (tfl) EXTERNAL STRING = "tfl";                              1B2AA51
    (tfnl) EXTERNAL STRING = "tfnl";                            1B2AA52
    (writedate) EXTERNAL STRING = "writedate";                  1B2AA53
    (readdate) EXTERNAL STRING = "readdate";                    1B2AA54
    (createdate) EXTERNAL STRING = "createdate";                1B2AA55
  % main branch identifiers %
    (libfile) EXTERNAL STRING = "LF:";                           1B2AB1
    (srcfil) EXTERNAL STRING = "SF:";                            1B2AB2
    (runfil) EXTERNAL STRING = "RF:";                            1B2AB3
    (newproc) EXTERNAL STRING = "NP:";                           1B2AB4
    (cmdbra) EXTERNAL STRING = "CB:";                            1B2AB5
    (libhis) EXTERNAL STRING = "Library History & Summaries";
                                                                 1B2AB6
  % builtin and user flag strings %
    (dindex) EXTERNAL = 0;                                       1B2AC1
    (nusrflgs) EXTERNAL CONSTANT = 40;                           1B2AC2
    (uflag) EXTERNAL STRING = "uf";                              1B2AC3
    (bvar1) EXTERNAL STRING = "filncmp";                         1B2AC4
    (bvar2) EXTERNAL STRING = "filcmp";                          1B2AC5
    (bvar3) EXTERNAL STRING = "oldfile";                         1B2AC6
    (bvar4) EXTERNAL STRING = "newfile";                         1B2AC7
    (bvar5) EXTERNAL STRING = "cmpok";                           1B2AC8
    (bvar6) EXTERNAL STRING = "cmperr";                          1B2AC9
    (bvar7) EXTERNAL STRING = "noerrors";                       1B2AC10
    (bvar8) EXTERNAL STRING = "totalerrors";                    1B2AC11
    (bvar9) EXTERNAL STRING = "nopc";                           1B2AC12
    (bvar10) EXTERNAL STRING = "filepc";                        1B2AC13
    (bvar11) EXTERNAL STRING = "infrun";                        1B2AC14
    (bvar12) EXTERNAL STRING = "infnrun";                       1B2AC15
    (bvar13) EXTERNAL STRING = "infok";                         1B2AC16
    (bvar14) EXTERNAL STRING = "infnok";                        1B2AC17
    (bvar15) EXTERNAL STRING = "numcompiles";                   1B2AC18
    (bvar16) EXTERNAL STRING = "numprints";                     1B2AC19
```

```
        (bvar17) EXTERNAL STRING = "numupdates";                1B2AC20
        (bvar18) EXTERNAL STRING = "numindexes";                1B2AC21
        (bvar19) EXTERNAL STRING = "numsysguides";              1B2AC22
        (bvar20) EXTERNAL STRING = "numconditionals";           1B2AC23
        (bvar21) EXTERNAL STRING = "numincludes";               1B2AC24
        (bvar22) EXTERNAL STRING = "numinferiors";              1B2AC25
        (bvar23) EXTERNAL STRING = "numrcompiles";              1B2AC26
        (bvar24) EXTERNAL STRING = "numicompiles";              1B2AC27
        (bvar25) EXTERNAL STRING = "numokinferiors";            1B2AC28
        (bvar26) EXTERNAL STRING = "numokcompiles";             1B2AC29
        (bvar27) EXTERNAL STRING = "sourcedate";                1B2AC30
        (bvar28) EXTERNAL STRING = "reldate";                   1B2AC31
        (bvar29) EXTERNAL STRING = "lpdate";                    1B2AC32
        (bvar30) EXTERNAL STRING = "lxdate";                    1B2AC33
        (bvar31) EXTERNAL STRING = "ludate";                    1B2AC34
        (bvar32) EXTERNAL STRING = "lidate";                    1B2AC35
        (bvar33) EXTERNAL STRING = "lndate";                    1B2AC36
        (bvar34) EXTERNAL STRING = "lcdate";                    1B2AC37
        (bvar35) EXTERNAL STRING = "libdate";                   1B2AC38
        (bvar36) EXTERNAL STRING = "inxdate";                   1B2AC39
        (bvar37) EXTERNAL STRING = "numpindices";               1B2AC40
        (bvar38) EXTERNAL STRING = "numpsysguide";              1B2AC41
        (bvar39) EXTERNAL STRING = "numlocked";                 1B2AC42
        (bvar40) EXTERNAL STRING = "anyaction";                 1B2AC43
        SET EXTERNAL nbnflgs= 40;   % number of builtin flags %
        (bvarstr) EXTERNAL = (                                  1B2AC45
           $bvar1, $filncmp,
           $bvar2, $filcmp,
           $bvar3, $oldfile,
           $bvar4, $newfile,
           $bvar5, $cmpok,
           $bvar6, $cmperr,
           $bvar7, $noerrors,
           $bvar8, $totalerrors,
           $bvar9, $nopc,
           $bvar10, $filepc,
           $bvar11, $infrun,
           $bvar12, $infnrun,
           $bvar13, $infok,
           $bvar14, $infnok,
           $bvar15, $numcompiles,
           $bvar16, $numprints,
           $bvar17, $numupdates,
           $bvar18, $numindexes,
           $bvar19, $numsysguides,
           $bvar20, $numconditionals,
           $bvar21, $numincludes,
           $bvar22, $numinferiors,
           $bvar23, $numrcompiles,
           $bvar24, $numicompiles,
           $bvar25, $numokinferiors,
           $bvar26, $numokcompiles,
           $bvar27, $sourcedate,
           $bvar28, $relfiledate,
           $bvar29, $lpdate,
           $bvar30, $lxdate,
```

```
            $bvar31, $ludate,
            $bvar32, $lidate,
            $bvar33, $lndate,
            $bvar34, $lcdate,
            $bvar35, $libdate,
            $bvar36, $inxdate,
            $bvar37, $numpindices,
            $bvar38, $numpsysguide,
            $bvar39, $numlocked,
            $bvar40, $anyaction );
        % flags %
          % the following are the command word driving variables and
          they are interpretted as follows:
            < 0: don't perform the specified action under any
            condition
            = 0: perform action if source file write date indicates
            its new
            > 0: perform action regardless of write date of source
            file %
          % builtin flags %
            (anyaction) EXTERNAL;              % 1 If any actions or
            errors %                                          1B2AD2A
            (numlocked) EXTERNAL;              % # files locked by
            someone else %                                    1B2AD2B
            (numpindices) EXTERNAL;  % # indices printed %     1B2AD2C
            (numpsysguide) EXTERNAL; % 1 if sysguide printed; 0
            otherwise %                                       1B2AD2D
            (lpdate) EXTERNAL;                 % date librarian last
            printed file %                                    1B2AD2E
            (lxdate) EXTERNAL;                 % date librarian last
            indexed file %                                    1B2AD2F
            (ludate) EXTERNAL;                 % date librarian last
            updated file %                                    1B2AD2G
            (lidate) EXTERNAL;                 % date librarian last
            counted includes %                                1B2AD2H
            (lndate) EXTERNAL;                 % date librarian last
            counted conditionals %                            1B2AD2I
            (lcdate) EXTERNAL;                 % date librarian last
            compiled file %                                   1B2AD2J
            (libdate) EXTERNAL;                % date librarian last
            run %                                             1B2AD2K
            (sourcedate) EXTERNAL;             % source file date %
                                                              1B2AD2L
            (relfiledate) EXTERNAL;  % rel file date %        1B2AD2M
            (inxdate) EXTERNAL;                % index file date %
                                                              1B2AD2N
            (numokcompiles) EXTERNAL;          % # successful compiles
            %                                                 1B2AD2O
            (numrcompiles) EXTERNAL; % # regular compiles done %
                                                              1B2AD2P
            (numicompiles) EXTERNAL; % # include compiles done %
                                                              1B2AD2Q
            (numcompiles) EXTERNAL;  % # compiles done %      1B2AD2R
            (numprints) EXTERNAL;              % # files printed %
                                                              1B2AD2S
            (numupdates) EXTERNAL;             % # files updated %
```

```
                                                                1B2AD2T
    (numindexes) EXTERNAL;          % # files indexed %
                                                                1B2AD2U
    (numsysguides) EXTERNAL; % # indices copied to sysguide
    %                                                           1B2AD2V
    (numconditionals) EXTERNAL;     % # files for which
    conditionals counted %                                      1B2AD2W
    (numincludes) EXTERNAL;  % # files for which includes
    counted %                                                   1B2AD2X
    (numinferiors) EXTERNAL; % # inferior forks run % 1B2AD2Y
    (numokinferiors) EXTERNAL;      % # inferior forks run
    successfully %                                              1B2AD2Z
    (infnok) EXTERNAL;              % NOT infok %   1B2AD2A@
    (infok) EXTERNAL;               % inferior ran ok %
                                                                1B2AD2AA
    (infnrun) EXTERNAL;             % NOT infrun %  1B2AD2AB
    (infrun) EXTERNAL;              % inferior has been run
    %                                                           1B2AD2AC
    (filepc) EXTERNAL;              % TRUE > file has a pc
    which is ours %                                             1B2AD2AD
    (nopc) EXTERNAL;          % NOT filepc %                  1B2AD2AE
    (totalerrors) EXTERNAL;  % total number of compile
    errors detected %                                           1B2AD2AF
    (noerrors) EXTERNAL;            % TRUE > no errors in
    any compile %                                               1B2AD2AG
    (cmperr) EXTERNAL;              % # errors during
    compile %                                                   1B2AD2AH
    (cmpok) EXTERNAL;               % TRUE > no errors
    during compile %                                            1B2AD2AI
    (newfile) EXTERNAL;             % TRUE > source more
    recent than rel %                                           1B2AD2AJ
    (oldfile) EXTERNAL;             % NOT newfile % 1B2AD2AK
    (filcmp) EXTERNAL;              % TRUE > file was
    compiled %                                                  1B2AD2AL
    (filncmp) EXTERNAL;             % NOT filcmp %  1B2AD2AM
(userflags) EXTERNAL [nusrflgs];    % user settable flags %
                                                                  1B2AD3
(dates) EXTERNAL [2];               % used for referencing
file dates %                                                     1B2AD4
(gabort) EXTERNAL;  % abort processing %                       1B2AD5
(grepeat) EXTERNAL; % repeat branch %                          1B2AD6
(sdrive) EXTERNAL;                                             1B2AD7
(grunfile) EXTERNAL;        % run an inferior %                1B2AD8
(gignore) EXTERNAL; % ignore this branch %                     1B2AD9
%(gdetach) EXTERNAL;        %% run detached if set -- not
in NLS10 %
(goutput) EXTERNAL; % > 0 implies jfn for redirected output
file %                                                         1B2AD11
(gsysgd) EXTERNAL;  % governs sysgds for individual files %
                                                               1B2AD12
(gconcnt) EXTERNAL; % governs conditional counts for
individual files %                                             1B2AD13
(gincent) EXTERNAL; % governs include counts for individual
files %                                                        1B2AD14
(gprint) EXTERNAL;  % governs printing of individual files
%                                                              1B2AD15
```

```
    (gptypescript) EXTERNAL;     % governs printing of
    typescript file %                                          1B2AD16
    (gplibbranch) EXTERNAL;      % governs printing of library
    branch %                                                   1B2AD17
    (gcopyprinter) EXTERNAL;  %governs copying of output to
    printer directory%                                         1B2AD18
    (gpindex) EXTERNAL; % governs printing of index files %
                                                               1B2AD19
    (gpsysguide) EXTERNAL;       % governs printing of sysguide
    file %                                                     1B2AD20
    (guptype) EXTERNAL; % governs updating type of individual
    files %                                                    1B2AD21
       % = 1: update old; = 2: update new; = 3: update compact
       %
    (gupdate) EXTERNAL; % governs updating of individual files
    %                                                          1B2AD22
    (gcompile) EXTERNAL;         % governs compiling of
    individual files %                                         1B2AD23
    (gsupersysgd) EXTERNAL;      % governs full sysgd generation
    %                                                          1B2AD24
    (glogout) EXTERNAL; % logout when done %                   1B2AD25
    %(gwait) EXTERNAL;           %% wait until time to run %
    (edrive) EXTERNAL;                                         1B2AD27
% constants for making x-routine arguments %
    (typwrtr) EXTERNAL CONSTANT = 52;  %should use declaration
    in (nine,psedt2)%                                          1B2AE1
    (dsply)EXTERNAL CONSTANT = 51;  %should use declaration in
    (nine,psedt2)%                                             1B2AE2
    (argxcw) EXTERNAL CONSTANT = 1;  %command word argument
    type %                                                     1B2AE3
    (argxvs) EXTERNAL CONSTANT = 2;  %viewspec argument type %
                                                               1B2AE4
%%                                                             1B2AF
```

```
% Procedures %
   (gtdate) PROCEDURE % get processing date from passed stid %
                                                                    1B3A
      % FORMALS %
        ( flstid);        % stid of concerned statement %    1B3A1A
      % LOCALS %
        LOCAL TEXT POINTER tp1, tp2;
        LOCAL STRING locstr[100];
      tp1 _ flstid;   tp1[1] _ 1;
      IF NOT FIND tp1 > [""(""] [":] ^tp1 < $PT > UL $3ULD CH $NP
      ^tp1 [""))""] < 2CH $NP ^tp2 THEN RETURN( 0);
      *locstr* _ tp1 tp2, 0;
      IF locstr.L <= 1 THEN RETURN( 0);
      IF NOT SKIP !idtim( 18M6 .V ($locstr + 1), 0) THEN R2 _ 0;
      RETURN( R2);
      END.
      %%                                                        1B3A10
```

```
(loadall) PROCEDURE % library init: load include & lib support
file %                                                          1B3B
   % FORMALS %
     ;
   % LOCALS %
     LOCAL ainclude;
     LOCAL TEXT POINTER tp1;
     LOCAL parray[40];
     (supload) _ FALSE;  %TRUE when loading support%          1B3B2D
     (retvalue) _ TRUE;                                       1B3B2E
        %return value -- set to FALSE if can't load include &
        lib support file%
     (emsg) STRING [2000];  %error message%                   1B3B2F
   INVOKE (catloadall, retloadall);
   IF NOT FIND SF( *upgnms*) ["INCLUDE"] THEN
     BEGIN
     dismes( 1, $"Loading INCLUDE Sequence Generator");
     tp1 _ $incfile;   tp1.stastr _ TRUE;
        tp1[1] _ 1;
     lnkprs( $tp1, $parray);
     ldprog( $parray, FALSE %no loader dismes%, $emsg);
     END;
   IF NOT FIND SF( *upgnms*) ["LIBSUP"] THEN
     BEGIN
     supload _ TRUE;
     dismes( 1, $"Loading LIBRARY Support Routines");
     tp1 _ $supfile;   tp1.stastr _ TRUE;
        tp1[1] _ 1;
     lnkprs( $tp1, $parray);
     ldprog( $parray, FALSE %no loader dismes%, $emsg);
     END;
   emsg.L _ 0;  %don't display undefines%
   (retloadall):                                             1B3B7
   DROP (catloadall);
   IF emsg.L THEN dismes(1, $emsg);  % display error message %
   RETURN( retvalue);
   (catloadall) CATCHPHRASE();                               1B3B11
     BEGIN
     CASE SIGNALTYPE OF
        = aborttype :
          BEGIN  %failure in loading%
          DISABLE (catloadall);
          IF supload THEN
             *emsg* _ "Unable to load LIBRARY Support
             Routines"
          ELSE *emsg* _ "Unable to load INCLUDE Sequence
          Generator";
          retvalue _ FALSE;  % return FALSE %
          TERMINATE;
          END;
        ENDCASE;
     CONTINUE;
     END;
   END.
   %%                                                        1B3B13
```

```
(loadchk) PROCEDURE % check to see if INCLUDE and SUPPORT
loaded %                                                        1B3C
   % FORMALS %
      ;
   % LOCALS %
   IF NOT FIND SF( *upgnms*) ["INCLUDE"] THEN
      BEGIN
      dismes( 1, $"User Program Buffer FOULUP; Please DELETE
      ALL programs & reload LIBRARY SUBSYS");
      RETURN( FALSE);
      END;
   IF NOT FIND SF( *upgnms*) ["LIBSUP"] THEN
      BEGIN
      dismes( 1, $"User Program Buffer FOULUP; Please DELETE
      ALL programs & reload LIBRARY SUBSYS");
      RETURN( FALSE);
      END;
   RETURN( TRUE);
   END.
   %%                                                           1B3C7
```

```
(xlbevtop) PROCEDURE % xroutine for process command %        1B3D
   % FORMALS %
      (destlist REF);  %branch entity%
   % LOCALS %
      LOCAL oldname;
      LOCAL oldmode;
      (destination) REF;                                     1B3D2C
      (emsg) REF;  %for abort error message%                 1B3D2D
      (arglist) LIST [2];  %argument for xsimulate%          1B3D2E
   % invoke catchphrase %
      INVOKE (catxlbevtop, retxlbevtop);
         % catch all ABORTS so that middle-end does not do an
         error return.  An error return would terminate the
         command and possibly leave the terminal in the wrong
         mode %
   % prepare parameters %
      &destination _ #destlist#[tppair];  %adr. of text ptr
      block%
   % do the command %
      IF NOT loadchk() THEN RETURN( FALSE);
      !gtad();
         libsrttime _ R1;      % library processing start time
         %
      !getnm();
         oldname _ R1;
      R1 _ 606354B6;  % sixbit for PSL %
         !setnm();
      gi _ 0;
      topalmost( destination);
      IF gi THEN
         BEGIN
         !dic( 4B5, 000010B6);
         %!iit( 4B5, 000010B6, 0);%
            R1 _ 4B5;   R2 _ 000010B6;   R3 _ 0; !JSYS 630B;
         chntab[14] _ gi;
         END;
      !setnm( oldname);
   (retxlbevtop):                                            1B3D6
   RETURN;
      (catxlbevtop) CATCHPHRASE(: &emsg);                    1B3D7A
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL
            = aborttype :
               BEGIN  %prevent middle-end from catching ABORT%

               DISABLE (catxlbevtop);
               IF &emsg AND emsg.L THEN dismes(1, &emsg)
               ELSE  %no error message -- tell user something%
                  dismes(1, $"Error in Process command -- no
                  explanation available");
               TERMINATE;  %goes to retxlbevtop%
               END;
            ENDCASE;
         CONTINUE;
```

```
        END;
    END.
%%
```

1B3D9

```
(xlchkadr)   % CL: ;  dummy xroutine for checking address %
PROCEDURE (source REF);                                          1B3F
   % Procedure description
      FUNCTION
         This procedure does not do anything.  It is called
         from the Process command so that argument conversion
         will check the address before the terminal mode gets
         changed.  If the address is not valid, this procedure
         will never be reached and the failure in the argument
         conversion procedure will cause the command to be
         aborted
      ARGUMENTS
         source:  address for library branch
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
   % Don't do anything %
   % Return %
    RETURN;
   END.

(xldelhis) PROCEDURE % xroutine for deleting history %     1B3F
   % FORMALS %
      (timelist REF LIST,      % text entity with time to
      delete before%
      destlist REF LIST);      % branch entity %
   % LOCALS %
      (destination) REF;  %adr. of text pointer block%   1B3F2A
      (timeptr) REF;  %adr. of text pointer block%       1B3F2B
      LOCAL histime;
      LOCAL hisstid;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING locstr [40];
   % prepare parameters %
      &destination _ #destlist#[tppair];
      &timeptr _ #timelist#[tppair];
   % do command %
      IF NOT loadchk() THEN RETURN( FALSE);
      % get internal date and time %
         tp1 _ timeptr;                  tp1[1] _ [&timeptr +
         1];
         tp2 _ [&timeptr + 2]; tp2[1] _ [&timeptr + 3];
         *locstr* _ tp1 tp2, 0;   BUMP DOWN locstr.L;
         IF NOT SKIP !idtim( 18M6 .V ($locstr + 1), 0) THEN
            BEGIN
            *locstr* _ *locstr*, " 12:00:00AM", 0;
            IF NOT SKIP !idtim( 18M6 .V ($locstr + 1), 0) THEN
               err( $"Illegal time specified");
            END;
         histime _ R2;
      % get stid of start %
```

```
            topstid _ destination;
         % take care of screen %
            clist( 15 %ctlcfm%, topstid.stfile, 0);
            dpset( dspstrc, topstid, endfil, dpstp(topstid));
         % loop through form deleting appropriately %
            cfstid _ getsub( topstid);
            WHILE cfstid # topstid DO
               BEGIN
               IF (srcstid _ getsub( cfstid)) # cfstid THEN
                  BEGIN
                  WHILE srcstid # cfstid DO
                     BEGIN
                     hisstid _ getsub( srcstid);
                     WHILE (hisstid # srcstid) AND (gtdate(
                     hisstid) >= histime) DO
                        hisstid _ getsuc( hisstid);
                     IF hisstid # srcstid THEN
                        cdelgro( hisstid, getail( hisstid), 0,
                        0);
                     srcstid _ getsuc( srcstid);
                     END;
                  END;
               cfstid _ getsuc( cfstid);
               END;
         % take care of screen %
            clupdt();
      RETURN;
      END.
      %%                                                  183F7
```

```
(xlicompile) PROCEDURE % xroutine compile file with includes %
                                                                1B3G
   %FORMALS%
      (entity REF,      %entity command word%
      location REF,     %selected location to begin%
      compname REF,     %name of compiler%
      flname REF,                          %name of output file%
      dname REF);       %diagnostic destination%
   LOCAL
      da,
      ret,
      savusq;  %save location of user sequence generator %
   LOCAL STRING
      locstr[100];
      (vs) [4];  %viewspec block%                               1B3G3B
   REF da, tptr2;
   % do command %
      IF NOT loadchk() THEN RETURN( FALSE);
      &da _ lda();
      inccount _ ret _ 0;
      savusq _ da.dausqcod := $include;
      INVOKE(catxlicompile, xicfix);
      makexarg(argxvs, $vs, $"0", da.dawid);
      ret _ xcompile( &entity, &location, &compname, &flname,
      &dname, $vs, da.dawid);
      (xicfix):                                                 1B3G5H
      DROP (catxlicompile);
      da.dausqcod _ savusq;
      CASE inccount OF
         = 0: *locstr* _ "No INCLUDEs encountered", CR, LF;
         ENDCASE
            *locstr* _ STRING( inccount), " INCLUDEs processed
            successfully", CR, LF;
      dismes( 1, $locstr);
   RETURN( ret );
   % catchphrases %
      (catxlicompile) CATCHPHRASE();                            1B3G7A
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               TERMINATE;
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                           1B3G9
```

```
(xliinit) PROCEDURE; % xroutine library initialization: load
include %                                                    1B3H
   % FORMALS %
   % LOCALS %
   % Load support and include sequence generator if necessary
   %
      IF NOT loaded THEN
         BEGIN
         IF NOT loadall() THEN RETURN( FALSE);
         loaded _ TRUE;
         END
      ELSE IF NOT loadchk() THEN RETURN( FALSE);
   RETURN;
   END.
   %%                                                        1B3H6
```

```
(xlreset)   % CL: ;  xroutine to reset the display %
PROCEDURE (dpytype REF, sourceptr REF);                          1B3I
   % Procedure description
      FUNCTION
         Call "xsimulate" to go back to display mode and
         "dpset" to refresh the screen.  This procedure is
         called from the Process command.
      ARGUMENTS
         dpytype:  DISPLAY command word
         sourceptr:  source pointer for Process command
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (source) REF;                                              1B3I2A
   % prepare argument %
      &source _ ELEM #sourceptr#[tppair];
   % Set up for screen refresh %
      dpset(dspallf, source, endfil, endfil);
   % Go back to display mode %
      xsimulate (&dpytype);
   % Return %
      RETURN;
   END.

   %%                                                            1B3I8
```

SKO, 12-Jul-78 15:33 < NINE, LIBRARY.NLS;28, > 20

FINISH

```
%libsup% FILE lsupport % (arcsubsys, 1109,)  (relnine, libsup.rel,)
%                                                                    1C
   NOMESS
   ALLOW!
   % DECLARATIONS %
      % strings for 'finder' -- can't be local because stack is
      small with sequence generator %
         (fdrstmnt) EXTERNAL STRING [2000];                          1C3A1
         (fdrstnum) EXTERNAL STRING [20];                            1C3A2
         (fdrcomment) EXTERNAL STRING [1500];                        1C3A3
         (fdrparams) EXTERNAL STRING [500];                          1C3A4
         (fdrtype) EXTERNAL STRING [50];                             1C3A5
         (fdrwidth) EXTERNAL STRING  [50];                           1C3A6
   (topalmost) PROCEDURE % process main branch %                       1C4
      % FORMALS %
         (libstid);                                                  1C4A1
      % LOCALS %
         LOCAL svint;
         LOCAL sgoutput;
         LOCAL i;
         LOCAL pcm;
         LOCAL init;
         LOCAL dirnum;
         LOCAL fl;   REF fl;
         LOCAL cp;
         LOCAL oldcp;
         LOCAL diff;
         LOCAL temp;
         LOCAL sav3;
         (emsg) REF;  %error message at ABORT%                       1C4B13
         LOCAL TEXT POINTER tp1, tp2;
         LOCAL darray[100];
         LOCAL STRING locstr[150];
         LOCAL STRING l2str[10];
         LOCAL STRING l3str[300];
         LOCAL STRING lkstr[100];
         LOCAL STRING messtr[1000];
      topstid _ libstid;
      svint _ FALSE;
      % initialize driving variables to default values %
         FOR i _ 0 UP UNTIL = nusrflgs DO userflags[i] _ -1;
         numlocked _ 0;                % 0 files locked by someone
         else %
         numpindices _ 0;     % 0 indices printed %
         numpsysguide _ 0;    % 0 sysguides printed %
         numcompiles _ 0;     % 0 compiles done %
         numokcompiles _ 0;   % 0 successful compiles done %
         numrcompiles _ 0;    % 0 regular compiles done %
         numicompiles _ 0;    % 0 include compiles done %
         numprints _ 0;                % 0 files printed %
         numupdates _ 0;      % 0 files updated %
         numindexes _ 0;      % 0 files indexed %
         numsysguides _ 0;    % 0 indices copied to sysguide %
         numconditionals _ 0;          % 0 files for which
         conditionals counted %
         numincludes _ 0;     % 0 files for which includes counted %
```

```
        numinferiors _ 0;    % 0 inferior forks run %
        numokinferiors _ 0; % 0 inferior forks run successfully %
        infrun _ -1;         % no inferiors run yet %
        infnrun _ - infrun;
        infok _ 1;  % inferior ran ok %
        infnok _ - infok;
        grunfile _ 1;
        totalerrors _ 0;
        anyaction _ 0;
        noerrors _ 1;
        gignore _ gabort _ grepeat _ -1;
        gsysgd _ 0; % do sysgd for each file %
        gconcnt _ 0;         % count conditionals for each file %
        gincent _ 0;         % count includes for each file %
        gprint _ 0; % print each file as appropriate %
        gpindex _ 0;         % print index files as appropriate %
        gptypescript _ 0;    % print typescript file as appropriate
        %
        gplibbranch _ 0;     % print library branch as appropriate %
        gcopyprinter _ 0;    % copy output to printer as appropriate
        %
        gpsysguide _ 0;      % print sysguide as appropriate %
        gsupersysgd _ 0;     % generate full sysgd %
        gcompile _ 0;        % gcompile each file %
        gupdate _ 0;         % default update each file %
        guptype _ 2;         % update new each file %
        %gdetach _ -1;       %% run attached --  detached not in
        NLS10 %
        glogout _ -1;        % dont logout when done %
        gwait _ 0;  % run immediately %
        !gpjfn( 400000B);    % get starting primary jfns %
           goutput _ srtjfns _ R2;
      % parse top statement - get stid for sysguide %
        tp2 _ topstid;   tp2[1] _ 1;
        IF NOT FIND tp2 > [*libfile*] ^tp2 THEN
           BEGIN
           dismes( 1, $"Invalid Plex header detected, Library
           processing aborted");
           RETURN( FALSE);
           END;
        IF NOT sprstid _ gmstid( topstid, cp _ tp2[1], TRUE, TRUE)
        THEN
           BEGIN
           dismes( 1, $"Invalid Plex header detected, Library
           processing aborted");
           RETURN( FALSE);
           END;
      % parse top statement - abort sysguide locked by some else %
        &fl _ flntadr( sprstid.stfile);
        IF fl.fllock AND (NOT fl.flpart) THEN
           BEGIN
           !gtfdb( fl.florig, 186 + 24B, $R3);
           init _ R3.lkinit;
           dirnum _ R3.lkdirn;
           lockid( $lkstr, dirnum, init);
           IF FIND SE(*lkstr*) ^tp2 [^(] SP 1$ULD ^tp1 THEN
```

```
            *lkstr* _ + tp1 tp2;
         *messtr* _
            ": LOCKED BY ", *lkstr*, "; Library processing
            aborted";
         wrtmsg( 0, $messtr);
         gsupersysgd _ -1;
         clsfile( sprstid := 0);
         RETURN( FALSE);
         END;
      % get some stids; abort if problems %
       *locstr* _ NULL;
       CASE crehis() OF
         < 0:
            *locstr* _ "Library Branch Substructure missing:
            Library processing aborted";
         = 0: NULL;
         > 0:
            *locstr* _ "Unable to create Library History Branch:
            Library processing aborted";
         ENDCASE;
       IF locstr.L THEN
         BEGIN
         dismes( 1,$locstr);
         IF sprstid THEN clsfile( sprstid := 0);
         RETURN( FALSE);
         END;
      % initialize printing file %
       IF NOT initprnt() THEN
         BEGIN
         dismes( 1, $"Unable to initialize printing file; Library
         processing aborted");
         IF sprstid THEN clsfile( sprstid := 0);
         RETURN( FALSE );
         END;
      % parse top statement - tell user we are starting %
       *locstr* _ "(( ))";
       tp1 _ $locstr;   tp1.stastr _ TRUE;   tp1[1] _ 1;
       rplpdate( tp1, $"Library processing started by");
       dismes( 1, $locstr);
       tp2 _ tp1;   tp2[1] _ locstr.L + 1;
       lhsstid _ cinssta( lhsstid, -1, $tp1, $tp2);
       ccopsta( lhsstid, -1, topstid, 0, 0);
      % parse top statement - initialize builtin flags %
       % get date librarian was last run %
         IF getftl( lhsstid) THEN libdate _ 0
         ELSE libdate _ gtdate( getsuc( lhsstid));
       % indicate whether or not we have a pc %
         filepc _ IF fl.flpart THEN 1 ELSE -1;
         nopc _ - filepc;
       % indicate any compile errors %
         cmperr _ -1;
         cmpok _ 1;
       % indicate if source newer than rel (ie has a pc in this
       case) %
         newfile _ filepc;
         oldfile _ - newfile;
```

```
        % indicate not compiled yet %
          filcmp _ -1;
          filncmp _ 1;
      % parse top statement - parse pre commands %
        IF NOT preparsecmd( topstid, oldcp _ cp : cp) THEN
          BEGIN
          dismes( 1, $"Unable to parse pre commands; Processing
          aborted");
          IF sprstid THEN clsfile( sprstid := 0);
          RETURN( FALSE);
          END;
        IF NOT sprstid THEN gsupersysgd _ -1;
        IF goutput # srtjfns THEN
          BEGIN
          jfntostr( goutput.RH, $13str, 011110B6 .V 1);
          *13str* _ "Changing primary output file to: ", *13str*;
          dismes( 1, $13str);
          % IF gdetach >= 0 THEN dismes( 1, $"Detaching job"); %
          !spjfn( 4000008, goutput);
          dismes( 1, $locstr);
          END;
        % IF gdetach >= 0 THEN
          BEGIN
          IF goutput = srtjfns THEN
            BEGIN
            dismes( 1, $"Illegal to detach without specifying an
            output file; detach command ignored");
            END
          ELSE
            BEGIN
            dismes( 1, $"Detaching job");
            !dtach();
            END;
          END; %
        IF oldcp # cp THEN
          BEGIN
          diff _ oldcp - cp;
          FOR temp _ $garray + 2 UP 2 UNTIL > $garray + 28 DO
            [temp] _ [temp] - diff;
          END;
          % wait not implemented in NLS10
        IF gwait THEN
          BEGIN
          sav3 _ chntab[3] := 1B6 .V $tclo;
          WHILE gwait > (5 * 60) DO
            BEGIN
            !dismS( (5 * 60) * 1000);
            gwait _ gwait - (5 * 60);
            END;
          !dismS( (gwait := 0) * 1000);
          !JFCL;
          (t2clo):                                              1C4L6F
          chntab[3] _ sav3;
          IF svint THEN
            BEGIN
            dismes( 1, $"Library processing aborted by user");
```

```
            glogout _ -1;
            GOTO done;
            END
         ELSE
            BEGIN
            rplpdate( tp1, $"Library processing resumed by");
            dismes( 1, $locstr);
            END;
         END;
      %
   % indicate no new file sysgds yet %
      INVOKE(cattopalmost1, top1);
      IF sprstid THEN isysgd( sprstid );
      newsysgdcnt _ 0;
   (top1):  % process each file %                                    1C4N
      DROP (cattopalmost1);
      sysorg _ srcstid _ 0;
      INVOKE (cattopalmost2, next);
      DISABLE (cattopalmost2);  %enable below%
      LOOP
         BEGIN
         % make listing nice %
            *l2str* _ CR, LF, CR, LF, CR, LF;
            dismes( 1, $l2str);
         % save driving variables state %
            pcm _ FALSE;
            FOR i _ $sdrive UP UNTIL >= $edrive DO
            darray[i-$sdrive] _ [i];
         ENABLE (cattopalmost2);
         % process the branch %
            tp2 _ cfstid;   tp2[1] _ 1;
            IF FIND tp2 > [*srcfil*] ^tp2 THEN dofile( tp2[1])
            ELSE IF FIND tp2 > [*runfil*] ^tp2 THEN runfork(
            tp2[1], FALSE)
            ELSE IF FIND tp2 > [*newproc*] ^tp2 THEN runfork(
            tp2[1], TRUE)
            ELSE IF FIND tp2 > [*cmdbra*] THEN pcmdbra( pcm _
            TRUE);
            DISABLE (cattopalmost2);
         (next):                                                  1C4N5F
         % close some files %
            IF sysorg THEN clsfile ( sysorg := 0);
            IF srcstid THEN clsfile( srcstid := 0);
         % restore driving variables %
            IF pcm := FALSE THEN
               BEGIN
               sgoutput _ darray[ $goutput - $sdrive];
               IF sgoutput # goutput THEN clspjfn( sgoutput.RH);
               END
            ELSE
               BEGIN
               sgoutput _ goutput;
               FOR i _ $sdrive UP UNTIL >= $edrive DO [i] _
               darray[i-$sdrive];
               IF sgoutput # goutput THEN
                  BEGIN
```

```
                !spjfn( 400000B, goutput);
                clspjfn( sgoutput.RH );
                END;
            END;
        % exit if ABORT asked for %
            IF gabort >= 0 THEN EXIT LOOP;
        % move on to next one if there is one (and no ^O) %
            IF (grepeat := -1) < 0 THEN
               BEGIN
               IF getftl( cfstid) THEN EXIT LOOP;
               cfstid _ getsuc( cfstid);
               END;
            IF svint OR (svint _ inptrf := FALSE) THEN EXIT LOOP;
        END;
(done):  % make listing look nice %                                  1C40
    DROP (cattopalmost2);
    *12str* _ CR, LF, CR, LF, CR, LF;
    dismes( 1, $12str);
    IF svint OR (svint _ inptrf := FALSE) THEN
       BEGIN
       IF gabort >= 0 THEN
          dismes( 1, $"Library processing ABORTED by command")
       ELSE dismes( 1, $"User terminated library processing");
       dismes( 1, $12str);
       END;
% finish up super sysgd if called or delete modifications %
    INVOKE (cattopalmost3, afsy);
    IF sprstid AND NOT (svint OR (svint _ inptrf := FALSE))
    THEN
       BEGIN
       fsuper();
       dismes( 1, $12str);
       END;
    (afsy):                                                          1C4P3
       DROP (cattopalmost3);
    IF sprstid THEN clsfile( sprstid := 0);
% set "anyaction" builtin flag %
    anyaction _
       IF numlocked OR numcompiles OR totalerrors OR
       numinferiors OR numupdates OR numprints OR numindexes OR
       numpindices OR numsysguides OR numpsysguide OR
       numincludes OR numconditionals
          THEN 1 ELSE 0;
% process any post commands %
    IF NOT postparsecmd(topstid, cp) THEN
       BEGIN
       dismes( 1, $"Unable to process post commands");
       dismes( 1, $12str);
       END;
% give user final summary %
    IF anyaction THEN
       BEGIN
       *messtr* _ NULL;
       IF numlocked THEN *messtr* _
          "***  ", STRING( numlocked), " FILE(S) LOCKED BY
          SOMEONE ELSE  ***", CR, LF;
```

```
        IF (i _ numcompiles - numokcompiles) OR totalerrors THEN
        *messtr* _
           *messtr*, "***  ", STRING( i), " UNSUCCESSFUL
           COMPILE(S); ", STRING( totalerrors), " TOTAL COMPILE
           ERROR(S)  ***", CR, LF;
        IF (i _ numinferiors - numokinferiors) THEN *messtr* _
        *messtr*,
           "***  ", STRING( i), " UNSUCCESSFUL INFERIOR
           PROCESS(ES)  ***", CR, LF;
        IF messtr.L THEN *messtr* _ *messtr*, CR, LF;
        CASE numupdates OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numupdates), " File(s) updated", CR, LF;
        CASE numcompiles OF
           = 0: NULL;
           = numrcompiles: *messtr* _ *messtr*,
              STRING( numcompiles), " Regular compile(s)", CR,
              LF;
           = numicompiles: *messtr* _ *messtr*,
              STRING( numcompiles), " Include compile(s)", CR,
              LF;
           ENDCASE *messtr* _ *messtr*,
              STRING( numcompiles), " Compile(s) (", STRING(
              numrcompiles), " Regular; ", STRING(
              numicompiles), " Include)", CR, LF;
        CASE numinferiors OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numinferiors), " Inferior process(es)
              run", CR, LF;
        CASE numprints OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numprints), " File(s) printed", CR, LF;
        CASE numindexes OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numindexes), " Index(es) created", CR, LF;
        CASE numpindices OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numpindices), " Index(es) printed", CR,
              LF;
        CASE numsysguides OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numsysguides), " Index(es) copied to
              sysguide", CR, LF;
        CASE numpsysguide OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              "Sysguide printed", CR, LF;
        CASE numincludes OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
```

```
            STRING( numincludes), " File(s) had its INCLUDEs
            counted", CR, LF;
         CASE numconditionals OF
            = 0: NULL;
            ENDCASE *messtr* _ *messtr*,
               STRING( numconditionals), " File(s) had its
               CONDITIONALs counted", CR, LF;
         END
      ELSE *messtr* _ "No actions performed", CR, LF;
      dismes( 1, $messtr);
      dismes( 1, $l2str);
      messtr.L _ messtr.L - 2;
         cnvcrlfteol( $messtr);
         tp1 _ $messtr;   tp1.stastr _ TRUE;   tp1[1] _ 1;
         tp2 _ tp1;   tp2[1] _ messtr.L + 1;
      cinssta( lhsstid, -1, $tp1, $tp2);
   % update library branch file & tell user %
      wrtmsg( -topstid.stfile, $": Updating (new)");
      INVOKE (cattopalmost4, done2);
      cupdfil( topstid.stfile, newversion, 0);
      wrtmsg( -topstid.stfile, $": Done Updating (new)");
      dismes( 1, $l2str);
      (done2):                                                  1C4T6
      DROP (cattopalmost4);
   IF NOT (svint OR (svint _ inptrf := 0)) THEN
      BEGIN
      % print the library branch %
         oplib();
      % copy file to arcprinter %
         rnfl();
         dismes( 1, $l2str);
      END;
   % tell user we are done %
      tp1 _ $locstr;   tp1.stastr _ TRUE;   tp1[1] _ 1;
      rplpdate( tp1, $"Library processing finished by");
      dismes( 1, $locstr);
   % restore starting primary jfns %
      IF goutput # srtjfns THEN
         BEGIN
         !spjfn( 400000B, srtjfns);
         svint _ inptrf := 0;
         clspjfn( goutput.RH);
         END;
   % logout if asked to %
      IF glogout >= 0 THEN xlogout();
   RETURN( TRUE);
   (tclo):                                                      1C4Z
      svint _ TRUE;
      chntab[3] _ sav3;
      lev1lc _ $t2clo;
      !debrk();
   % catchphrases %
      (cattopalmost1) CATCHPHRASE();                            1C4A@1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
```

```
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost1);
               dismes( 1, $"Unable to initialize sysgd file,
               Sysguide functions ABORTED");
               gsupersysgd _ gpsysguide _ -1;
               IF sprstid THEN clsfile( sprstid := 0);
               TERMINATE;  %goes to 'topl'%
               END;
            ENDCASE;
         CONTINUE;
         END;
      (cattopalmost2) CATCHPHRASE( : &emsg);                    1C4A@2
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost2);
               IF NOT (svint _ inptrf := FALSE) THEN
                  BEGIN  % not ^O %
                  IF emsg.L THEN wrtmsg( 0, emsg);
                  wrtmsg( 0, $": *** Error while processing");
                  END;
               TERMINATE;  %goes to 'next'%
               END;
            ENDCASE;
         CONTINUE;
         END;
      (cattopalmost3) CATCHPHRASE();                            1C4A@3
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost3);
               wrtmsg( -sprstid.stfile, $": Unable to finish
               processing sysguide");
               dismes( 1, $12str);
               TERMINATE;  %goes to 'afsy'%
               END;
            ENDCASE;
         CONTINUE;
         END;
      (cattopalmost4) CATCHPHRASE( : emsg );                    1C4A@4
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost4);
               IF emsg THEN dismes( 1, emsg);
```

```
            dismes( 1, $12str);
            TERMINATE;  %goes to 'done2'%
            END;
         ENDCASE;
      CONTINUE;
      END;
END.
%%
```

1C4AB

```
(clsfile) PROCEDURE % close file we are done with %                    1C5
   % FORMALS %
     ( flstid);  % stid of origin of concerned file %                  1C5A1
   % LOCALS %
   IF NOT flstid.stfile THEN RETURN;
   INVOKE (catclsfile, retclsfile);
   close( flstid.stfile);
   (retclsfile):                                                       1C5F
   DROP (catclsfile);
   RETURN;
   % catchphrases %
     (catclsfile) CATCHPHRASE();                                       1C5I1
       BEGIN
       CASE SIGNALTYPE OF
          = notetype : NULL;
          = helptype : NULL;
          = aborttype :
          ENDCASE;
             TERMINATE;
       CONTINUE;
       END;
   END.
   %%                                                                  1C5K
```

```
(clspjfn) PROCEDURE % copy to <*prtdir*> and close a primary
output jfn %                                                    1C6
   % FORMALS %
      (ojfn);                                                   1C6A1
   % LOCALS %
      LOCAL njfn; % jfn of printer copy %
      LOCAL lchar;
      LOCAL tnread;
      LOCAL nread;
      LOCAL bytecnt;
      LOCAL byteptr;
      LOCAL STRING locstr[500];
   % dont copy unless asked for %
      IF gptypescript < 0 THEN
         BEGIN
         IF NOT SKIP !closf( ojfn) THEN NULL;
         RETURN( TRUE);
         END;
   jfntostr( ojfn, $locstr, 001100B6 .V 1);
   *locstr* _ "<, *prtdir*, ">, *initsr*, "-, *locstr*,
   ";P777777", 0;
   IF NOT SKIP !gtjfn( 4B11 .V 1B6, 18M6 .V ($locstr + 1) ) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      RETURN( FALSE );
      END;
   njfn _ R1;
   IF NOT SKIP !openf( njfn, 07B10 .V3B5) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( FALSE );
      END;
   IF NOT SKIP !rfptr( ojfn) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( FALSE );
      END;
   bytecnt _ R2;
   IF NOT SKIP !sfptr( ojfn, 0) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( FALSE );
      END;
   tnread _ 0;
   WHILE tnread < bytecnt DO
      BEGIN
      !sin( ojfn, 18M6 .V ($locstr + 1), 499, 37B);
      byteptr _ R2;
      nread _ 499 - R3;
      tnread _ tnread + nread;
      lchar _ .byteptr;
      IF lchar = 37B THEN
         BEGIN
```

```
            lchar _ 15B;
            .byteptr _ lchar;
            lchar _ 12B;^byteptr _ lchar;
            BUMP nread;
            END;
         !sout( njfn, 18M6 .V ($locstr + 1), -nread);
         END;
      IF NOT SKIP !delnf( ojfn, 2) THEN NULL;
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( TRUE);
      END.
      %%                                                    1C6S
```

```
(cmpfile) PROCEDURE % compile file if needed %                          1C7
   % FORMALS %
      ;
   % LOCALS %
      LOCAL savvs;
      LOCAL savvs2;
      LOCAL savca;
      LOCAL savus;
      LOCAL savcsp;
      LOCAL da;   REF da;
      LOCAL errors;
      LOCAL nincludes;
      (vsptr) _ 0;  %set to $vs if INCLUDE's%                          1C7B9
      (retvalue) _ 1 ;                                                 1C7B10
      (vs) [4];  %viewspec block%                                      1C7B11
      LOCAL TEXT POINTER tp1, tp2, ta1, ta2;
      LOCAL STRING locstr[200];
   % find out if we need compile %
      CASE gcompile OF
         < 0: RETURN( 0);
         = 0: IF reldate >= srcwrd THEN RETURN( 0);
         ENDCASE;
   % find out if any includes %
      tp1 _ incstid;   tp1[1] _ 1;
      nincludes _ 0;
      IF FIND tp1 > $NP ^tp1 1$D ^tp2 THEN
         BEGIN
         *locstr* _ tp1 tp2;
         nincludes _ VALUE( $locstr );
         END;
   % tell user which file we are starting to compile %
      IF nincludes > 0 THEN *locstr* _ ": Include compiling"
      ELSE *locstr* _ ": Compiling";
      wrtmsg( 0, $locstr);
   % find out and set tp1 to where to start compiling %
      &da _ lda();
      tp1 _ srcstid;   tp1[1] _ 1;
      ta1 _ aarray[ls];   ta1[1] _ aarray[ls+1];
      ta2 _ aarray[le];   ta2[1] _ aarray[le+1];
      caddexp( $ta1, $ta2, &da, $tp1)  %sets tp1%;
      DROP (catcmpfile1);
   % do it %
      errors _ 0;
      savcsp _ da.dacsp := tp1;
      savca _ da.dacacode := 0;
      savus _ da.dausqcode := 0;
      savvs2 _ da.davspc2 := 0;
      savvs _ da.davspec := 0;
         da.davspec.vslev _ da.davspec.vstrnc _ da.davspec.vsindf
         _ da.davspec.vsnamf _ -1;
      IF nincludes > 0 THEN
         BEGIN
         da.dausqcod _ $include;
         makexarg(argxvs, $vs, $"0", da.dawid);  %form vs arg%
         vsptr _ $vs;  %use sg vs%
         inccount _ 0;
```

```
        END;
      INVOKE (catcmpfile2, ecmp);
      errors _ ccompile(cwfile, tp1, $cmpptr, $relptr, 0, vsptr,
      &da  );
      (ecmp):  DROP (catcmpfile2);;                                  1C7G10
      da.dacsp _ savcsp;
      da.davspec _ savvs;
      da.davspc2 _ savvs2;
      da.dacacode _ savca;
      da.dausgcod _ savus;
    % tell user of errors or completion %
      IF nincludes > 0 THEN BUMP numicompiles ELSE BUMP
      numrcompiles;
      BUMP numcompiles;
      filcmp _ 1;   filncmp _ -1;
      IF errors <= 0 THEN
         BEGIN
         *locstr* _ ": Done ", *locstr*[3 TO locstr.L];
         BUMP numokcompiles;
         END
      ELSE
         BEGIN
         totalerrors _ errors + totalerrors;
         noerrors _ -1;
         cmperr _ errors;   cmpok _ -1;
         *locstr* _ ": ", STRING( errors), " errors detected
         while ", *locstr*[3 TO locstr.L];
         END;
      wrtmsg( 0, $locstr);
      IF nincludes > 0 THEN
         BEGIN
         *locstr* _ ": ";
         IF inccount > 0 THEN *locstr* _
            *locstr*, STRING( inccount), " INCLUDEs processed
            successfully"
         ELSE *locstr* _ *locstr*, "No INCLUDEs encountered";
         wrtmsg( 0, $locstr);
         END;
    % update relstid with [error] count %
      ccopsta( relstid, -1, relstid, FALSE, FALSE);
      *locstr* _ NULL;
      IF errors > 0 THEN *locstr* _ STRING( errors), " errors; ";
      IF nincludes THEN *locstr* _ *locstr*, "Include ";
      *locstr* _ *locstr*, "Compiled by";
      rpipdate( relstid, $locstr);
      ccopsta( lhsstid, -1, relstid, 0, 0);
    (retcmpfile):                                                    1C7J
    DROP (catcmpfile2);
    RETURN( retvalue );
    % catchphrases %
      (catcmpfile1) CATCHPHRASE();                                   1C7M1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
```

```
            ENDCASE;
               BEGIN
               DISABLE (catcmpfile1);
               wrtmsg( 0, $": Unable to determine compiling start
               address, compile aborted");
               retvalue _ 0;
               TERMINATE;  %goes to "retcmpfile"%
               END;
         CONTINUE;
         END;
      (catcmpfile2) CATCHPHRASE();                                  1C7M2
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
            ENDCASE;
               BEGIN
               DISABLE (catcmpfile2);
               BUMP errors;
               TERMINATE;  %goes to "ecmp"%
               END;
         CONTINUE;
         END;
   END.
   %%                                                               1C70
```

```
(cplex) PROCEDURE % get stids of [created] stmnts under file
driving stmnt %                                                    1C8
   % FORMALS %
      ;
   % LOCALS %
      LOCAL TEXT POINTER tp1, tp2, fp1, fp2, fp3, fp4;
      LOCAL STRING flname[200];
      LOCAL STRING locstr[300];
   CASE (procstid _ getsub( cfstid)) OF
      = cfstid:   % must create the statements %
         BEGIN
         % create compile statement next %
            fp1 _ fp2 _ fp3 _ fp4 _ garray[us];
               fp1[1] _ garray[us+1];
               fp2[1] _ garray[ue+1];
               fp3[1] _ garray[fs+1];
               fp4[1] _ garray[fe+1];
               IF FIND fp4 < [";] ^fp4 THEN NULL;
               IF FIND fp4 < [".] ^fp4 THEN NULL;
            *locstr* _ "Compile <";
               IF fp1[1] # fp2[1] THEN *locstr* _ *locstr*, -fp1
               fp2, ",;
               *locstr* _ *locstr*, -fp3 fp4, ",1> Using ",
               *compiler*, " To <";
               IF fp1[1] # fp2[1] THEN *locstr* _ *locstr*, -fp1
               fp2, ",;
               *locstr* _ *locstr*, -fp3 fp4, ".rel,>";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            relstid _ cinssta( cfstid, -1, $tp1, $tp2);
         % create conditional statement next %
            *locstr* _ "Conditionals;";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            constid _ cinssta( cfstid, -1, $tp1, $tp2);
         % create include statement first %
            *locstr* _ "Includes;";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            incstid _ cinssta( cfstid, -1, $tp1, $tp2);
         % create index statement next %
            fp1 _ fp2 _ fp3 _ fp4 _ garray[us];
               fp1[1] _ garray[us+1];
               fp2[1] _ garray[ue+1];
               fp3[1] _ garray[fs+1];
               fp4[1] _ garray[fe+1];
               IF FIND fp4 < [";] ^fp4 THEN NULL;
               IF FIND fp4 < [".] ^fp4 THEN NULL;
            *locstr* _ "<;
               IF fp1[1] # fp2[1] THEN *locstr* _ *locstr*, -fp1
```

```
            fp2, ",";
            *locstr* _ *locstr*, "index-", -fp3 fp4, ",>";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         sysstid _ cinssta( cfstid, -1, $tp1, $tp2);
      % create print history statement next %
         *locstr* _ "Print History";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         prntstid _ cinssta( cfstid, -1, $tp1, $tp2);
      % create update history statement next %
         *locstr* _ "Update History";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         uphstid _ cinssta( cfstid, -1, $tp1, $tp2);
      % create processing history statement next %
         *locstr* _ "Processing History";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         procstid _ cinssta( cfstid, -1, $tp1, $tp2);
      END;
   ENDCASE     % presumably they exist %
      BEGIN
      IF getftl( procstid) THEN REPEAT CASE( cfstid)
      ELSE uphstid _ getsuc( procstid);
      IF getftl( uphstid) THEN REPEAT CASE( cfstid)
      ELSE prntstid _ getsuc( uphstid);
      IF getftl( prntstid) THEN REPEAT CASE( cfstid)
      ELSE sysstid _ getsuc( prntstid);
      IF getftl( sysstid) THEN REPEAT CASE( cfstid)
      ELSE incstid _ getsuc( sysstid);
      IF getftl( incstid) THEN REPEAT CASE( cfstid)
      ELSE constid _ getsuc( incstid);
      IF getftl( constid) THEN REPEAT CASE( cfstid)
      ELSE relstid _ getsuc( constid);
      END;
RETURN;
END.
%%                                                    1C8F
```

```
(cqplex) PROCEDURE % get stid history stmnt under CB/RF/NP
driving stmnt %                                                    1C9
   % FORMALS %
     ;
   % LOCALS %
     LOCAL TEXT POINTER tp1, tp2;
     LOCAL STRING locstr[100];
   CASE (procstid _ getsub( cfstid)) OF
     = cfstid:   % must create the statements %
        BEGIN
        % create processing history statement next %
          *locstr* _ "Processing History";
          tp1 _ tp2 _ $locstr;
             tp1.stastr _ tp2.stastr _ TRUE;
             tp1[1] _ 1;
             tp2[1] _ locstr.L + 1;
          procstid _ cinssta( cfstid, -1, $tp1, $tp2);
        END;
     ENDCASE;     % presumably it exists %
   RETURN;
   END.
   %%                                                              1C9F
```

```
(cpysplex) PROCEDURE % copy plex from sysgd to supersysgd %   1C10
   % FORMALS %
      ;
   % LOCALS %
      LOCAL plsstid;
      LOCAL plestid;
   IF (NOT sprstid) OR (NOT sysorg) THEN RETURN;
   wrtmsg( 0, $": Copying index to sysguide");
   IF (plsstid _ getnxt( sysorg)) # endfil THEN
      BEGIN
      plestid _ getail( plsstid);
      copgrp( sprstid, -1, plsstid, plestid, FALSE);
      BUMP numsysguides;
      END;
   RETURN;
   END.
   %%                                                    1C10H
```

```
(crehis) PROCEDURE % get lib history  & 1st branch stids %     1C11
   % FORMALS %
     ;
   % LOCALS %
     (retvalue) ;                                               1C11B1
     LOCAL TEXT POINTER tp1, tp2;
   IF (lhsstid _ getsub( topstid)) = topstid THEN RETURN( -1);
   tp1 _ lhsstid;   tp1[1] _ 1;
   IF NOT FIND tp1 > [*libhis*] THEN
      BEGIN
      tp1 _ $libhis;   tp1.stastr _ TRUE;   tp1[1] _ 1;
      tp2 _ tp1;   tp2[1] _ libhis.L + 1;
      INVOKE (catcrehis, retcrehis);
      cfstid _ lhsstid := cinssta( topstid, -1, $tp1, $tp2);
      retvalue _ 0;
      END
   ELSE
      IF (cfstid _ getsuc( lhsstid)) = topstid THEN
         retvalue _ - 1
      ELSE retvalue _ 0;
   (retcrehis);                                                 1C11G
   DROP (catcrehis);
   RETURN( retvalue );
   % catchphrases %
      (catcrehis) CATCHPHRASE();                                1C11J1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               retvalue _ 1;
               TERMINATE;
               END;
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                           1C11L
```

```
(dofile) PROCEDURE % process one file %                              1C12
   % FORMALS %
      (cp);        % char count in cfstid after *srcfile* %        1C12A1
   % LOCALS %
      LOCAL fl;   REF fl;
      LOCAL didsomething;
      LOCAL dirnum;
      LOCAL init;
      LOCAL oldcp;
      LOCAL diff;
      LOCAL temp;
      LOCAL savgoutput;
      LOCAL savpgoutput;
      LOCAL TEXT POINTER tp1, tp2, tp3;
      LOCAL STRING l3str[300];
      LOCAL STRING lkstr[100];
      LOCAL STRING messtr[100];
      LOCAL STRING locstr[2000];
      LOCAL STRING stacop[2000];
   % get copy of driving statement for later copying %
      tp1 _ cfstid;   tp1[1] _ 1;
      *stacop* _ tp1 SE( tp1);
   % indicate nothing done intitially %
      didsomething _ 0;
   % get stid for source file and sub-plex %
      IF NOT srcstid _ gmstid( cfstid, cp, TRUE, FALSE ) THEN
         BEGIN
         wrtmsg( 0, $": unable to parse link or load file for
         this branch, branch ignored");
         RETURN;
         END;
      &fl _ flntadr( srcstid.stfile);
      IF fl.fllock AND (NOT fl.flpart) THEN
         BEGIN
         BUMP numlocked;
         !gtfdb( fl.florig, 1B6 + 24B, $R3);
         init _ R3.lkinit;
         dirnum _ R3.lkdirn;
         lockid( $lkstr, dirnum, init);
         IF FIND SE(*lkstr*) ^tp2 [^(] SP 1$ULD ^tp1 THEN
            *lkstr* _ + tp1 tp2;
         *messtr* _
            ": LOCKED BY ", *lkstr*, "; Processing started";
         END
      ELSE *messtr* _ ": Processing started";
      wrtmsg( 0, $messtr);
      cplex();
   % get write date for this file %
      sourcedate _ srcwrd _ getwrtdate( srcstid);
   % get compile parameters %
      INVOKE (catdofile, retdofile);
      relfiledate _ reldate _ getcompile();
      DISABLE(catdofile);  %dropped later%
   % initialize builtin flags %
      % get dates librarian last performed action %
         lpdate _ gtdate( prntstid);
```

```
            lxdate _ gtdate( sysstid);
            ludate _ gtdate( uphstid);
            lidate _ gtdate( incstid);
            lndate _ gtdate( constid);
            lcdate _ gtdate( relstid);
         % get stid and write date for index file %
            IF NOT sysorg _ gmstid( sysstid, 1, FALSE, TRUE) THEN
            inxdate _ 0
            ELSE inxdate _ getwrtdate( sysorg);
         % indicate whether or not we have a pc %
            filepc _ IF fl.flpart THEN 1 ELSE -1;
            nopc _ - filepc;
         % indicate any compile errors %
            cmperr _ 0;
            cmpok _ 1;
         % indicate if source newer than rel %
            newfile _ IF srcwrd > reldate THEN 1 ELSE -1;
            oldfile _ - newfile;
         % indicate not compiled yet %
            filcmp _ -1;
            filncmp _ 1;
      % parse any pre commands (and handle output redirection) %
         savgoutput _ goutput;
         IF NOT preparsecmd( cfstid, oldcp _ cp : cp ) THEN
            BEGIN
            wrtmsg( cfstid, $": unable to parse commands for this
            file, branch ignored");
            RETURN;
            END;
         IF gabort >= 0 THEN RETURN;
         IF gignore >= 0 THEN
            BEGIN
            wrtmsg( 0, $": File ignored");
            RETURN;
            END;
         IF savgoutput # goutput THEN
            BEGIN
            jfntostr( goutput.RH, $13str, 011110B6 .V 1);
            *13str* _ "Changing primary output file to: ", *13str*;
            dismes( 1, $13str);
            !spjfn( 400000B, goutput);
            END;
         IF NOT sprstid THEN gsupersysgd _ -1;
         IF oldcp # cp THEN
            BEGIN
            diff _ oldcp - cp;
            FOR temp _ $garray + 2 UP 2 UNTIL > $garray + 28 DO
               [temp] _ [temp] - diff;
            END;
      % update the file if asked for (maybe) %
         CASE gupdate OF
            < 0: NULL;
            = 0: IF srcwrd = 35M THEN REPEAT CASE( gupdate _ 1);
            > 0: didsomething _ didsomething + upfile();
            ENDCASE;
      % print file, gen & update & print sysgd, count (maybe) %
```

```
      didsomething _ didsomething + prntfile();
   % compile file (maybe) %
      didsomething _ didsomething + cmpfile();
   % process any post commands %
      savpgoutput _ goutput;
      IF NOT postparsecmd( cfstid, cp ) THEN
         BEGIN
         wrtmsg( 0, $": unable to parse post commands for this
         file");
         RETURN;
         END;
      IF savpgoutput # goutput THEN
         BEGIN
         jfntostr( goutput.RH, $13str, 011110B6 .V 1);
         *13str* _ "Changing primary output file to: ", *13str*;
         dismes( 1, $13str);
         !spjfn( 400000B, goutput);
         IF savpgoutput # savgoutput THEN clspjfn(
         savpgoutput.RH);
         END;
      IF gabort >= 0 THEN RETURN;
   % indicate file processed %
      IF lkstr.L THEN *messtr* _ "LOCKED BY ", *lkstr*, "; "
      ELSE *messtr* _ NULL;
      *messtr* _ *messtr*, "Processed by";
      tp1 _ tp2 _ $stacop;
         tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ stacop.L + 1;
         tp3 _ cfstid;   tp3[1] _ 1;
            *locstr* _ tp3 SE( tp3);
         IF (*locstr* # *stacop*) THEN BUMP didsomething;
      IF didsomething THEN
         BEGIN
         cinssta( procstid, -1, $tp1, $tp2);
         rplpdate( cfstid, $messtr);
         END;
   wrtmsg( 0, $": Done processing");
   (retdofile):                                                  1C12P
   DROP (catdofile);
   RETURN;
   % catchpharases %
      (catdofile) CATCHPHRASE();                                 1C12S1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (catdofile);
               wrtmsg( 0,$": Unable to determine compile
               parameters, branch ignored");
               TERMINATE;
               END;
            ENDCASE;
         CONTINUE;
```

```
        END;
    END.
%%
```

1C12U

```
(fndqual) PROCEDURE % find first qualifier %                    1C13
   % FORMALS %
      ;
   % LOCALS %
      LOCAL invert;
      LOCAL var1, var2, m;
      LOCAL TEXT POINTER tp1;
   invert _ 2;
   IF FIND *never* $NP THEN invert _ -1
   ELSE IF FIND *default* $NP THEN invert _ 0
   ELSE IF FIND *always* $NP THEN invert _ 1
   ELSE IF FIND *tft* $NP THEN
      invert _ IF userflags[ guindex()] > 0 THEN 1 ELSE -1
   ELSE IF FIND *tff* $NP THEN
      invert _ IF userflags[ guindex()] < 0 THEN 1 ELSE -1
   ELSE IF FIND *tfm* $NP THEN
      invert _ IF userflags[ guindex()] = 0 THEN 1 ELSE -1
   ELSE IF FIND *tfnt* $NP THEN
      invert _ IF userflags[ guindex()] <= 0 THEN 1 ELSE -1
   ELSE IF FIND *tfnf* $NP THEN
      invert _ IF userflags[ guindex()] >= 0 THEN 1 ELSE -1
   ELSE IF FIND *tfnm* $NP THEN
      invert _ IF userflags[ guindex()] # 0 THEN 1 ELSE -1
   ELSE IF FIND *tfe* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 = var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfne* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 # var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfg* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 > var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfng* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
```

```
      );
      invert _ IF var1 <= var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfl* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 < var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfnl* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 >= var2 THEN 1 ELSE -1;
      END;
   RETURN( IF invert # 2 THEN TRUE ELSE FALSE, invert);
   END.
   %%                                                    1C13U
```

```
(fsuper) PROCEDURE % final processing super sysgd file %      1C14
   % FORMALS %
      ;
   % LOCALS %
      LOCAL temp;
      LOCAL TEXT POINTER tp1;
   IF NOT sprstid THEN RETURN;
   CASE gsupersysgd OF
      < 0:
         BEGIN
         cdelmodfil( sprstid.stfile);
         wrtmsg( -sprstid.stfile, $": Modifications deleted");
         numsysguides _ 0;
         CASE gpsysguide OF
            = 0:
               BEGIN
               tp1 _ topstid;   tp1[1] _ 1;
               IF FIND tp1 > ["Printed by"] THEN temp _ gtdate(
               topstid)
               ELSE temp _ 0;
               IF (temp = 0) OR (temp < getwrtdate(sprstid)) THEN
                  REPEAT CASE(1);
               END;
            > 0:
               BEGIN
               oqsysgd( sprstid, FALSE);
               rplpdate( topstid, $"Sysguide Printed by");
               END;
            ENDCASE;
         END;
      = 0: IF newsysgdcnt = 0 THEN REPEAT CASE( -1) ELSE REPEAT
      CASE( 1);
      > 0:
         BEGIN
         upsysgd( sprstid);
         IF gpsysguide >= 0 THEN
            BEGIN
            oqsysgd( sprstid, FALSE);
            rplpdate( topstid, $"Sysguide Created & Printed by");
            END
         ELSE rplpdate( topstid, $"Sysguide Created by");
         RETURN;
         END;
      ENDCASE;
   IF gpsysguide > 0 THEN oqsysgd( sprstid, FALSE);
   RETURN;
   END.
   %%                                                  1C14H
```

```
(getcompile) PROCEDURE % get rel filename and date and compiler
name %                                                          1C15
   % FORMALS %
     ;
   % LOCALS %
     LOCAL jfn;
     LOCAL etirwdate;
     LOCAL TEXT POINTER tp1, tp2;
   % parse address link first %
     tp1 _ relstid;   tp1[1] _ 1;
     lnkprs( $tp1, $aarray);
   % now parse the compile link and set text pointer to it %
     tp1 _ aarray[le];   tp1[1] _ aarray[le+1];
     lnkprs( $tp1, $carray);
     cmpptr _ carray[ls];
     cmpptr[1] _ carray[ls + 1];
   % now get rel file name and set text pointer to it %
     tp1 _ carray[le];   tp1[1] _ carray[le+1];
     lnkprs( $tp1, $carray);
     relptr _ carray[ls];
     relptr[1] _ carray[ls + 1];
     lnbfls( $relptr, 0, $relname);
     *relname* _ *relname*, 0;  %need it in TENEX format for
     JSYS%
     BUMP DOWN relname.L;
   % see if rel file exists %
     IF NOT SKIP !gtjfn( 1B11 .V 1B6, 18M6 .V ($relname + 1))
     THEN RETURN( 0);
     jfn _ R1;
   % it does so get its etirwdate and release jfn %
     !gtfdb( jfn, 1B6 .V 14B, $etirwdate);
     IF NOT SKIP !rljfn( jfn) THEN NULL;
   RETURN( etirwdate);
   END.
   %%                                                           1C15J
```

```
(getfnum) PROCEDURE % read number from command %                    1C16
   % FORMALS %
      ;
   % LOCALS %
      LOCAL minus;
      LOCAL val;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING locstr[50];
   minus _ FALSE;
   IF FIND "- $NP THEN minus _ TRUE;
   IF FIND ^tp1 1$D ^tp2 $NP THEN
      BEGIN
      *locstr* _ tp1 tp2;
      val _ VALUE($locstr);
      IF minus THEN val _ - val;
      END
   ELSE val _ 0;
   RETURN( val);
   END.
   %%                                                              1C16I
```

```
(getwrtdate) PROCEDURE % get write date for passed stid %      1C17
   % FORMALS %
      ( flstid);  % stid of concerned file %                  1C17A1
   % LOCALS %
      LOCAL etirwdate;
      LOCAL fl; REF fl;
   &fl _ flntadr( flstid.stfile);
   IF fl.flpart THEN etirwdate _ 35M
   ELSE !gtfdb( fl.florig, 1B6 .V 14B, $etirwdate);
   RETURN( etirwdate);
   END.
   %%                                                         1C17H
```

```
(gmstid) PROCEDURE % get stid of origin statement for link at
passed stid %                                                    1C18
   % FORMALS %
      ( flstid,    % stid of statement containing link %
      cp,          % start position for link search %
      mode,              % TRUE: copy parsed link array to
      global area %
      crefl);      % if file doesn't exist: TRUE: create; FALSE:
      error %
   % LOCALS %
      LOCAL rhostn;
      (retvalue);                                                1C18B2
      (abortsw) _ FALSE;                                         1C18B3
      LOCAL parray[40];
      LOCAL TEXT POINTER tp1, tp2, tp4, rp1;
      LOCAL STRING locstr[100];
      LOCAL STRING filstr[200];
   % parse link thereby finding its delimeters %
      tp1 _ tp2 _ tp4 _ rp1 _ flstid;
         tp1[1] _ tp4[1] _ cp;
      INVOKE (catgmstid, badretgmstid);  %if 'lnkprs' does ABORT%
      lnkprs( $tp1, $parray);
      DROP (catgmstid);
      IF mode THEN FOR mode _ 0 UP UNTIL = 40 DO garray[mode] _
      parray[mode];
      tp1[1] _ parray[ls+1];
      tp2[1] _ rp1[1] _ parray[le+1];
   INVOKE (catgmstid, gmscre);  %if 'caddexp' does ABORT%
   caddexp( $tp1, $tp2, lda(), $tp4);
   (gmscre):                                                     1C18F
   DROP (catgmstid);
   IF abortsw THEN
      BEGIN  %ABORT occured%
      IF inptrf THEN
         err($"User terminated processing");  %ABORT from ^O%
      IF crefl THEN  %ABORT in 'caddexp'%
         BEGIN  % file does not exist -- create file%
         INVOKE (catgmstid, badretgmstid);  %in case create
         fails%
         rhostn _ lnbfls( 0, $parray, $filstr);
         tp4 _ ccrefil( rhostn, $filstr);
         DROP (catgmstid);
         tp4.stpsid _ origin;
         retvalue _ tp4;  %set return value%
         END
      ELSE retvalue _ FALSE;  %file does not exist -- don't
      create%
      END
   ELSE
      BEGIN  %file exists%
      tp4.stpsid _ origin;
      retvalue _ tp4;  %set return value%
      END;
   RETURN( retvalue, rp1[1]);
   (badretgmstid):  %here from catchphrase%                      1C18K
      IF inptrf THEN err($"User terminated processing");
```

```
      RETURN( FALSE);
   %catchphrases%
      (catgmstid) CATCHPHRASE();                                    1C18L1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN  %file exists%
               abortsw _ TRUE;
               TERMINATE;
                  %This catchphrase is INVOKE'd and DROP'd
                  several times.  For the first INVOKE,  the
                  TERMINATE goes to 'retgmstid', for the second
                  INVOKE,  the TERMINATE goes to 'gmscre' and for
                  the third INVOKE the TERMINATE goes to
                  'retgmstid' %
               END;
            ENDCASE;
         CONTINUE;
         END;
      (catgmstid1) CATCHPHRASE();                                   1C18L2
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               TERMINATE;
                  %goes to 'retgmstid' for first INVOKE and to
                  'gmscre'  for second INVOKE %
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                               1C18N
```

```
(guindex) PROCEDURE % get index for flag array %                    1C19
   % FORMALS %
      ;
   % LOCALS %
      LOCAL n, dflag, tjfn, tdate;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING nstring[200];
   IF FIND *uflag* ^tp1 1$D ^tp2 $PT $NP THEN
      BEGIN
      *nstring* _ tp1 tp2;
      n _ VALUE( $nstring );
      IF n NOT IN [1, nusrflgs] THEN n _ 0;
      RETURN( n);
      END;
   FIND ^tp1 $PT ^tp2 $NP;
   *nstring* _ tp1 tp2;
   FOR n _ 0 UP UNTIL >= $nbnflgs DO
      IF *nstring* = *[bvarstr[2*n]]* THEN
         RETURN( bvarstr[(2*n)+1] - $userflags);
   dflag _ 0;
   IF *nstring* = *writedate* THEN dflag _ 14B
   ELSE IF *nstring* = *readdate* THEN dflag _ 15B
   ELSE IF *nstring* = *createdate* THEN dflag _ 13B;
   IF dflag > 0 THEN
      BEGIN
      FIND ^tp1 $PT ^tp2 $NP;
      *nstring* _ tp1 tp2, 0;
      dindex _ dindex .X 1;
      n _ $dates + dindex - $userflags;
      userflags[n] _ 0;
      IF SKIP !gtjfn( 100001B6, 18M6 .V ($nstring + 1) ) THEN
         BEGIN
         tjfn _ R1;
         !gtfdb( tjfn, 1B6 .V dflag, $tdate);
         userflags[n] _ tdate;
         IF NOT SKIP !rljfn( tjfn) THEN NULL;
         RETURN( n);
         END;
      END;
   RETURN( 0 );
   END.
   %%                                                               1C19N
```

```
(initprnt) PROCEDURE % initialize printing file %                    1C20
   % FORMALS %
      ;
   % LOCALS %
      LOCAL fl, fjfn;
      LOCAL jfn;
      LOCAL page;
      REF fl;
   INVOKE (catinitprnt, badretinitprnt);
   &fl _ flntadr( sprstid.stfile);
   IF (fjfn _ fl.florig) = 0 THEN RETURN( FALSE);
   jfntostr( fjfn, $qpfilename, 001000B6);
   *qpfilename* _ *qpfilename*, ".LIB-LIST", 0;
   BUMP DOWN qpfilename.L;
   DROP (catinitprnt);
   IF NOT SKIP !gtjfn(4B11 .V 1B6, 18M6 .V ($qpfilename + 1))
   THEN
      RETURN( FALSE);
   jfn _ R1;
   IF NOT SKIP !openf( jfn, 7B10 .V3B5) THEN
      BEGIN
      IF NOT SKIP !rljfn( jfn) THEN NULL;
      RETURN( FALSE);
      END;
   page _ 0;   page.LH _ jfn;
   LOOP
      BEGIN
      IF NOT SKIP !ffufp( page) THEN EXIT LOOP;
      page _ R1;
      !pmap( -1, page, 0);
      BUMP page;
      END;
   IF NOT SKIP !sfptr( jfn, 0) THEN NULL;
   IF NOT SKIP !closf( jfn) THEN RETURN( FALSE);
   RETURN( TRUE);
   (badretinitprnt):  %here from catchphrase%                        1C20R
      DROP (catinitprnt);
      RETURN (FALSE);
   % catchphrases %
      (catinitprnt) CATCHPHRASE();                                   1C20S1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               TERMINATE;  %GOES TO "BADRETINITPRNT" %
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                                1C20U
```

```
(isysgd) PROCEDURE % initialize a sysgd file %                    1C21
   % FORMALS %
      ( flstid);  % stid of origin of a sysgd file %             1C21A1
   % LOCALS %
      LOCAL plstid;
      LOCAL plestid;
   IF NOT flstid THEN RETURN;
   % update it first %
      upsysgd( flstid);
   % make sure plex 1 exists %
      plstid _ ccopsta( flstid, -1, flstid, FALSE, 0);
   % now delete plex 1 %
      plestid _ getail( plstid);
      cdelgro( plstid, plestid, 0, 0);
   RETURN;
   END.
   %%                                                            1C21I
```

```
(makexarg)   % CL: ;  make argument for an x-routine %
PROCEDURE (argtype, argvar REF, element1, element2, element3,
element4, element5, element6, element7, element8, element9,
element10);                                                    1C22
   % Procedure description
      FUNCTION
         Create an argument for an x-routine in the format that
         comes out of the argument conversion routine
      ARGUMENTS
         argtype:  type of argument
            value: argxcw -- command word
                   argxvs -- viewspec
         argvar:  adr. of variable for converted argument
            command word -- this is a list
            viewspec -- this is a 4 word array
         element1, element2, ..., element 10:  element value,
         dependent on argtype value
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (vsblk) REF;  %adr. vs block%                            1C22B1
      (vs) LIST [1]; %list for vs arg%                         1C22B2
   % check argument type and fill argument list %
      CASE argtype OF
         = argxcw:  %command word%
            BEGIN
            IF element1 THEN
               #argvar# _ element1  %token command value%
            ELSE #argvar# _ NULL;
            IF element2 THEN
               #argvar# !_ *[element2]*  %command word%
            ELSE #argvar# !_ NULL;
            END;
         = argxvs:  %viewspec%
            BEGIN
            #vs# _ cnvvsp (element1, element2);  %convert vs
            string%
            &vsblk _ (ELEM #vs#[1]).RH;  %adr. vs block%
            argvar _ vsblk;
            argvar[1] _ vsblk[1];  %copy into passed array%
            argvar[2] _ vsblk[2];
            argvar[3] _ vsblk[3];
            #vs# _ ;  %null out list%
            END;
         ENDCASE;
   % Return %
      RETURN;
      %%                                                       1C22D2
```

```
    END.

(nxtqual) PROCEDURE % find next qualifier %                          1C23
    % FORMALS %
      ( invert);                                                     1C23A1
    % LOCALS %
      LOCAL opor;
      LOCAL opand;
      LOCAL opxor;
      LOCAL ninvert;
      LOCAL var1;
      LOCAL var2;
    opand _ opor _ opxor _ FALSE;
    IF FIND ". ( "o FS opor / "a FS opand / "x FS opxor) $NP THEN
      IF fndqual( : ninvert) THEN
        BEGIN
        var1 _ CASE invert OF
          < 0: 0;
          = 0: 3;
          ENDCASE 6;
        var2 _ CASE ninvert OF
          < 0: 0;
          = 0: 1;
          ENDCASE 2;
        CASE TRUE OF
          = opor:  invert _  ortable[var1 + var2];
          = opand: invert _ andtable[var1 + var2];
          = opxor: invert _ xortable[var1 + var2];
          ENDCASE;
        RETURN( TRUE, invert);
        END;
    RETURN( FALSE, invert);
    END.
    %%                                                               1C23G
```

```
(oplib) PROCEDURE % print the library branch %                    1C24
   % FORMALS %
      ;
   % LOCALS %
      LOCAL da;   REF da;
      LOCAL savvs;
      LOCAL savvs2;
      LOCAL savus;
      LOCAL savca;
      LOCAL savoqapfg;
      LOCAL savcsp;
   IF gplibbranch < 0 THEN RETURN;
   &da _ lda();
   savcsp _ da.dacsp := topstid;
   savoqapfg _ oqapfg := TRUE;
   savca _ da.dacacode := 0;
   savus _ da.dausgcode := 0;
   savvs2 _ da.davspc2 := 0;
   savvs _ da.davspec := 0;
      da.davspec.vslev _ getlev( topstid) + 2;
      da.davspec.vstrnc _ -1;
      da.davspec.vsindf _ da.davspec.vsnamf _ TRUE;
      da.davspec.vsblkf _ da.davspec.vsbrof _ TRUE;
   opseqf( $gpfilename, &da, FALSE, TRUE, opgpfl);
   da.dacsp _ savcsp;
   oqapfg _ savoqapfg;
   da.dacacode _ savca;
   da.dausgcode _ savus;
   da.davspc2 _ savvs2;
   da.davspec _ savvs;
   RETURN;
   END.
   %%                                                            1C24T
```

```
(oqsysgd) PROCEDURE % print a sysgd / index %                    1C25
   % FORMALS %
      ( flstid,    % origin stid sysgd file %
      mode);       % TRUE: file index; FALSE: super sysgd %
   % LOCALS %
      LOCAL da;    REF da;
      LOCAL savqpcolmax;
      LOCAL savvs;
      LOCAL savvs2;
      LOCAL savus;
      LOCAL savca;
      LOCAL savoqapfg;
      LOCAL savcsp;
   IF mode THEN BUMP numpindices ELSE BUMP numpsysguide;
   wrtmsg( -flstid.stfile, $": Printing");
   &da _ lda();
   savqpcolmax _ uqpcolmax := 80;
   savcsp _ da.dacsp := IF mode THEN getnxt( flstid) ELSE flstid;
   savoqapfg _ oqapfg := TRUE;
   savca _ da.dacacode := 0;
   savus _ da.dausqcode := 0;
   savvs2 _ da.davspc2 := 0;
   savvs _ da.davspec := 0;
      da.davspec.vslev _ da.davspec.vstrnc _ IF mode THEN 1 ELSE
      -1;
      da.davspec.vsindf _ da.davspec.vsnamf _ TRUE;
      IF NOT mode THEN da.davspec.vsblkf _ TRUE;
   IF da.dacsp # endfil THEN opseqf( $qpfilename, &da, FALSE,
   TRUE, opqpfl);
   uqpcolmax _ savqpcolmax;
   da.dacsp _ savcsp;
   oqapfg _ savoqapfg;
   da.dacacode _ savca;
   da.dausqcode _ savus;
   da.davspc2 _ savvs2;
   da.davspec _ savvs;
   wrtmsg( -flstid.stfile, $": Done printing");
   RETURN;
   END.
   %%                                                           1C25X
```

```
(pcmdbra) PROCEDURE % process one command branch statement %  1C26
   % FORMALS %
     ;
   % LOCALS %
     LOCAL sgoutput;
     LOCAL STRING l3str[300];
   dismes( 1, $"Processing command statement");
   cgplex();
   ccopsta( procstid, -1, cfstid, FALSE, FALSE);
   sgoutput _ goutput;
   IF NOT postparsecmd( cfstid, 1) THEN
      BEGIN
      dismes( 1, $"*** Error while processing branch");
      END;
   IF sgoutput # goutput THEN
      BEGIN
      jfntostr( goutput.RH, $l3str, 011110B6 .V 1);
      *l3str* _ "Changing primary output file to: ", *l3str*;
      dismes( 1, $l3str);
      !spjfn( 400000B, goutput);
      END;
   rpipdate( cfstid, $"Processed by");
   dismes( 1, $"Processing done");
   RETURN;
   END.
   %%                                                    1C26M
```

```
(postparsecmd) PROCEDURE % parse driving statement post commands
%                                                              1C27
   % FORMALS %
     (sstid,     % stid of statement with commands %
     cp);                % char position after *srcfile* %
   % RETURNS %
     % returns FALSE on error %
   % LOCALS %
     LOCAL savel;
     LOCAL TEXT POINTER cps, cpe;
     LOCAL STRING locstr[1500];
   cpe _ cps _ sstid;   cps[1] _ cp;   cpe[1] _ 1;
   IF NOT FIND cps > [""<<" / "##"] < 2CH ^cps SE( cps) ^cpe THEN
     RETURN( TRUE);
   *locstr* _ cps cpe;
   savel _ locstr.L;
   IF NOT prepost( $locstr) THEN RETURN( FALSE);
   IF savel # locstr.L THEN ST cps cpe _ *locstr*;
   RETURN( TRUE);
   END.
   %%                                                          1C27L
```

```
(prcfile) PROCEDURE % do file using output quickprint / sequence
generator %                                                    1C28
   % FORMALS %
      ;
   % LOCALS %
      LOCAL da;   REF da;
      LOCAL seqa;   REF seqa;
      LOCAL savvs;
      LOCAL savvs2;
      LOCAL savus;
      LOCAL savca;
      LOCAL savoqapfg;
      LOCAL savcsp;
      LOCAL myvspec;
      LOCAL myvspc2;
   % set up and save da record, viewspecs, etc. %
      &da _ lda();
      savcsp _ da.dacsp := srcstid;
      savoqapfg _ oqapfg := TRUE;
      savca _ da.dacacode := $finder;
      savus _ da.dausqcode := 0;
      savvs2 _ da.davspc2 := myvspc2 _ 0;
      savvs _ da.davspec := 0;
         da.davspec.vslev _ da.davspec.vstrnc _ da.davspec.vscapf
         _ da.davspec.vsindf _ da.davspec.vsnamf _
         da.davspec.vsstnf _ da.davspec.vsstnr _ -1;
      myvspec _ da.davspec;
   IF gprint > 0 THEN opseqf( $qpfilename, &da, FALSE, TRUE,
   opqpfl)
   ELSE
      BEGIN
      &seqa _ operseq( srcstid, seqend( srcstid, myvspec,
      myvspc2), myvspec, myvspc2, 0, $finder);
      WHILE (seqgen( &seqa) # endfil) AND (NOT inptrf) DO NULL;
      closeseq( &seqa);
      END;
   % restore da record, etc. %
      da.dacsp _ savcsp;
      oqapfg _ savoqapfg;
      da.dacacode _ savca;
      da.dausqcode _ savus;
      da.davspc2 _ savvs2;
      da.davspec _ savvs;
   RETURN;
   END.
   %%                                                          1C28I
```

```
(preparsecmd) PROCEDURE % parse driving statement pre commands %
                                                                  1C29
   % FORMALS %
     (sstid,      % stid of statement with commands %
     cp);                  % char position after *srcfile* %
   % RETURNS %
     % returns FALSE on error %
   % LOCALS %
     LOCAL savel;
     LOCAL TEXT POINTER cps, cpe;
     LOCAL STRING locstr[1500];
   cpe _ cps _ sstid;   cps[1] _ 1;   cpe[1] _ cp;
   IF NOT FIND cpe < [">>" / "##"] > 2CH ^cpe THEN RETURN( TRUE,
   cp);
   *locstr* _ cps cpe;
   savel _ locstr.L;
   IF NOT prepost( $locstr) THEN RETURN( FALSE);
   IF savel # locstr.L THEN
      BEGIN
      ST cps cpe _ *locstr*;
      cp _ cp - (savel - locstr.L);
      END;
   RETURN( TRUE, cp);
   END.
   %%                                                             1C29L
```

```
(prepost) PROCEDURE % parse pre/post commands %                    1C30
   % FORMALS %
      (astr);     % adr string containing commands %              1C30A1
      REF astr;
   % RETURNS %
      % returns FALSE on error %
   % LOCALS %
      LOCAL permc;
      LOCAL cmd;
      LOCAL rstid;
      LOCAL TEXT POINTER tp1, cp1, cp2, cp3, cp4;
   % point to start of string %
      tp1 _ &astr;
         tp1.stastr _ TRUE;
         tp1[1] _ 1;
   % look at what comes next (permanent or temp commands /
   nothing) %
      FIND tp1;
      LOOP
         BEGIN
         FIND > [(("<<" FS permc / "##" FR permc) FS cmd ^cp2 <
         2CH $NP ^cp1) / ENDCHR FR cmd];
         IF NOT cmd THEN EXIT LOOP;
         IF permc THEN
            IF NOT FIND cp2 > [2$">] ^cp4 < 2CH ^cp3 THEN
               BEGIN
               FIND cp2;
               REPEAT LOOP;
               END
            ELSE NULL
         ELSE
            IF NOT FIND cp2 > [2$"#] ^cp4 < 2CH ^cp3 THEN
               BEGIN
               FIND cp2;
               REPEAT LOOP;
               END
            ELSE NULL;
         rdcmd( $cp2, $cp3);
         IF permc THEN FIND cp4
         ELSE
            BEGIN
            ST cp1 cp4 _ NULL;
            FIND cp1;
            END;
         END;
   RETURN( TRUE);
   END.
   %%                                                             1C30H
```

```
(prntfile) PROCEDURE % print file; gen, update, print sysgd;
count; maybe %                                                    1C31
   % FORMALS %
     ;
   % LOCALS %
     LOCAL todo;
     LOCAL indxdate;
     LOCAL lpxdate;
     LOCAL fl; REF fl;
     LOCAL TEXT POINTER tp2, tp3;
     LOCAL STRING locstr[300];
   % get stid and write date for index file %
     IF NOT sysorg THEN
        BEGIN
        wrtmsg( 0, $": Unable to load index file; Index
        functions aborted");
        gpindex _ gsupersysgd _ gsysgd _ -1;
        indxdate _ 35M;
        END
     ELSE
        BEGIN
        &fl _ flntadr( sysorg.stfile);
        IF fl.fllock AND (NOT fl.flpart) THEN
           BEGIN
           BUMP numlocked;
           wrtmsg( 0, $": Index file locked by someone else;
           Index creation aborted");
           gsysgd _ -1;
           indxdate _ 35M;
           END
        ELSE IF (indxdate _ inxdate) = 35M THEN indxdate _ 0;
        END;
   % see if anything to do %
     todo _ FALSE;
     CASE gprint OF
        < 0: NULL;
        = 0:
           IF srcwrd > gtdate( prntstid) THEN gprint _ todo _ 1
           ELSE gprint _ -1;
        > 0: todo _ 1;
        ENDCASE;
     CASE gconcnt OF
        < 0: NULL;
        = 0:
           IF srcwrd > gtdate( constid) THEN gconcnt _ todo _ 1
           ELSE gconcnt _ -1;
        > 0: todo _ 1;
        ENDCASE;
     CASE gincent OF
        < 0: NULL;
        = 0:
           IF srcwrd > gtdate( incstid) THEN gincent _ todo _ 1
           ELSE gincent _ -1;
        > 0: todo _ 1;
        ENDCASE;
     CASE gsysgd OF
```

```
            < 0: NULL;
            = 0:
               IF (srcwrd > indxdate) OR ((indxdate >= libsrttime)
               AND (indxdate # 35M)) THEN
                  gsysgd _ todo _ 1
               ELSE gsysgd _ -1;
            > 0: todo _ 1;
            ENDCASE;
         IF sysorg AND (gsysgd < 0) THEN
            BEGIN
            IF gsupersysgd >= 0 THEN cpysplex();
            CASE gpindex OF
               = 0:
                  BEGIN
                  tp2 _ sysstid;   tp2[1] _ 1;
                  lpxdate _
                     IF FIND tp2 > ["Printed by"] THEN lxdate ELSE
                     0;
                  CASE lpxdate OF
                     = 0: REPEAT CASE 2( 1);
                     < indxdate: IF indxdate # 35M THEN REPEAT CASE
                     2( 1);
                     ENDCASE;
                  END;
               > 0:
                  BEGIN
                  oqsysgd( sysorg, TRUE);
                  rplpdate( sysstid, $"Printed by");
                  clsfile( sysorg := 0);
                  IF NOT todo THEN RETURN( 1);
                  END;
               ENDCASE;
            END;
         IF NOT todo THEN
            BEGIN
            IF sysorg THEN clsfile( sysorg :=0);
            RETURN( 0);
            END;
      % get stid for index file origin %
         IF sysorg THEN
            IF gsysgd < 0 THEN clsfile( sysorg := 0)
            ELSE
               BEGIN
               isysgd( sysorg);
               filenum _ srcstid.stfile;
               *fname* _ NULL;
               filnam( filenum, $fname);
               FIND SF( *fname* ) ["<"] $NP ^tp2 ["."] < CH ^tp3;
               *fname* _ "<", - tp2 tp3, ",";
               END;
      % initialize counters, etc. %
         condcnt _ incldcnt _ filenum _ infile _ inproc _ inrecord _
         0;
      % tell user what we are about to do %
         *locstr* _ NULL;
         IF gprint > 0 THEN *locstr* _ "Printing";
```

```
      IF gsysgd > 0 THEN
         BEGIN
         IF locstr.L THEN *locstr* _ *locstr*, ", ";
         *locstr* _ *locstr*, "Generating index";
         END;
      IF gconcnt > 0 THEN
         BEGIN
         IF locstr.L THEN *locstr* _ *locstr*, ", ";
         *locstr* _ *locstr*, "Counting CONDITIONALs";
         END;
      IF ginccnt > 0 THEN
         BEGIN
         IF locstr.L THEN *locstr* _ *locstr*, ", ";
         *locstr* _ *locstr*, "Counting INCLUDEs";
         END;
      *locstr* _ ": ", *locstr*;
      wrtmsg( 0, $locstr);
   % use either output quickprint or our own sequence generator %
      prcfile();
      IF gprint > 0 THEN
         BEGIN
         ccopsta( prntstid, -1, prntstid, FALSE, FALSE);
         rplpdate( prntstid, $"Printed by");
         END;
   % update counters in driver file %
      IF gconcnt > 0 THEN upconcnt();
      IF ginccnt > 0 THEN upinccnt();
   % tell user we are done %
      *locstr* _ ": Done ", *locstr*[3 TO locstr.L];
      wrtmsg( 0, $locstr);
   % update, print, copy sysgd file %
      IF sysorg AND (NOT inptrf) THEN
         BEGIN
         upsysgd( sysorg);
         ccopsta( sysstid, -1, sysstid, FALSE, FALSE);
         IF gpindex >= 0 THEN ogsysgd( sysorg, TRUE);
         rplpdate( sysstid, IF gpindex >= 0 THEN $"Created &
         Printed by" ELSE $"Created by");
         BUMP newsysgdcnt;
         IF gsupersysgd >= 0 THEN cpysplex();
         END;
   % update builtin counters %
      IF gsysgd > 0 THEN BUMP numindexes;
      IF gconcnt > 0 THEN BUMP numconditionals;
      IF ginccnt > 0 THEN BUMP numincludes;
      IF gprint > 0 THEN BUMP numprints;
   RETURN( 1);
   END.
   %%                                                    1C310
```

```
(rdcmd) PROCEDURE % interpret one command %                    1C32
   % FORMALS %
      ( cps,       % adr text pointer start of command %
      cpe);        % adr text pointer end of command %
   REF cps, cpe;
   % LOCALS %
      LOCAL n;
      LOCAL m;
      LOCAL var1;
      LOCAL var2;
      LOCAL minus;
      LOCAL jfn;
      LOCAL invert;
      LOCAL minutes;
      LOCAL hours;
      LOCAL seconds;
      LOCAL wfor;
      LOCAL wuntil;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING 12str[20];
      LOCAL STRING locstr[1500];
   FIND cps > $NP ^tp1 cpe < $NP ^tp2;
   *locstr* _ - tp1 tp2;
   FIND SF(*locstr*) ^tp1;
   IF fndqual( : invert) THEN WHILE nxtqual( invert : invert) DO
   NULL
   ELSE invert _ 1;
   FIND > $NP ^tp1;
   IF FIND *csever* THEN
      BEGIN
      gprint _ gcompile _ gsysgd _ gsupersysgd _ ginccnt _
      gconcnt _ gupdate _ gptypescript _ gplibbranch _ gpindex _
      gpsysguide _ invert;
      RETURN;
      END;
   IF FIND *cslist* THEN
      BEGIN
      gprint _ gptypescript _ gplibbranch _ gpindex _ gpsysguide
      _ invert;
      RETURN;
      END;
   IF FIND *cscomp* THEN
      BEGIN
      gcompile _ invert;
      RETURN;
      END;
   IF FIND *cspfil* THEN
      BEGIN
      gprint _ invert;
      RETURN;
      END;
   IF FIND *cssysg* THEN
      BEGIN
      gsupersysgd _ invert;
      RETURN;
      END;
```

```
IF FIND *csrunf* THEN
   BEGIN
   grunfile _ invert;
   RETURN;
   END;
IF FIND *csindx* THEN
   BEGIN
   gsysgd _ invert;
   RETURN;
   END;
IF FIND *csupdt* THEN
   IF invert < 0 THEN
      BEGIN
      gupdate _ -1;
      RETURN;
      END
   ELSE
      BEGIN
      IF (gupdate _ invert) >= 0 THEN
         IF FIND $NP "old" THEN guptype _ 1
         ELSE IF FIND $NP "new" THEN guptype _ 2
         ELSE IF FIND $NP "compact" THEN guptype _ 3;
      RETURN;
      END;
IF FIND *cscond* THEN
   BEGIN
   gconcnt _ invert;
   RETURN;
   END;
IF FIND *csincl* THEN
   BEGIN
   ginccnt _ invert;
   RETURN;
   END;
IF FIND *csptyp* THEN
   BEGIN
   gptypescript _ invert;
   RETURN;
   END;
IF FIND *csplib* THEN
   BEGIN
   gplibbranch _ invert;
   RETURN;
   END;
IF FIND *cscopyprinter* THEN
   BEGIN
   gcopyprinterbranch _ invert;
   RETURN;
   END;
IF FIND *cspind* THEN
   BEGIN
   gpindex _ invert;
   RETURN;
   END;
IF FIND *cspsys* THEN
   BEGIN
```

```
        gpsysguide _ invert;
        RETURN;
        END;
    %IF FIND *csdtch* THEN
        BEGIN
        gdetach _ invert;
        RETURN;
        END; %
    IF FIND *cscomm* $NP ^tp1 THEN
        BEGIN
        IF invert >= 0 THEN
            BEGIN
            *locstr* _ tp1 SE(*locstr*);
            dismes( 1, $locstr);
            END;
        RETURN;
        END;
    IF FIND *csstat* $NP ^tp1 THEN
        BEGIN
        IF invert >= 0 THEN
            BEGIN
            n _ guindex();
            IF n = 0 THEN RETURN;
            FIND ^tp2;
            *locstr* _ "Status of ", + tp1 tp2, " is: ",
                *[( CASE userflags[n] OF
                    < 0: $"FALSE (";
                    = 0: $"MAYBE (";
                    ENDCASE $"TRUE (" )]*,
                *[( CASE userflags[n] OF
                    < 0: $never;
                    = 0: $default;
                    ENDCASE $always )]*,
                ");
            dismes( 1, $locstr);
            END;
        RETURN;
        END;
    IF FIND *csoctl* $NP ^tp1 THEN
        BEGIN
        IF invert >= 0 THEN
            BEGIN
            n _ guindex();
            IF n = 0 THEN RETURN;
            FIND ^tp2;
            *locstr* _
                "Octal value of ", + tp1 tp2, " is: ",  STRING(
                userflags[n], 8);
            dismes( 1, $locstr);
            END;
        RETURN;
        END;
    IF FIND *csdcml* $NP ^tp1 THEN
        BEGIN
        IF invert >= 0 THEN
            BEGIN
```

```
            n _ guindex();
            IF n = 0 THEN RETURN;
            FIND ^tp2;
            *locstr* _
               "Decimal value of ", + tp1 tp2, " is: ",  STRING(
               userflags[n]);
            dismes( 1, $locstr);
            END;
         RETURN;
         END;
      IF FIND *csdate* $NP ^tp1 THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            n _ guindex();
            IF n = 0 THEN RETURN;
            FIND ^tp2;
            CASE userflags[n] OF
               = 35M:
                  BEGIN
                  !odtim( 18M6 .V ($12str + 1), -1, 0);
                  12str.L _ 12str.M;
                     n _ 0;
                     WHILE *12str*[n+1] DO BUMP n;
                     12str.L _ n;
                  END;
               = 0: *12str* _ "NEVER";
               ENDCASE
                  BEGIN
                  !odtim( 18M6 .V ($12str + 1), userflags[n], 0);
                  12str.L _ 12str.M;
                     n _ 0;
                     WHILE *12str*[n+1] DO BUMP n;
                     12str.L _ n;
                  END;
            *locstr* _
               "Date value of ", + tp1 tp2, " is: ", *12str*;
            dismes( 1, $locstr);
            END;
         RETURN;
         END;
      IF FIND *csoutp* THEN
         BEGIN
         % This is the command to change the primary output file.
         It should result in a call on the FE.  This FE feature has
         not yet been implemented.  Therefore, the command will be
         ignored and an error message will be displayed to the user.
         %
         *locstr* _ "Feature not implemented: command ignored",
            CR, LF, cps cpe;
         dismes(1, $locstr);
         RETURN;
         END;
      IF FIND *for* $NP THEN
         BEGIN
         IF invert >= 0 THEN
```

```
            BEGIN
            var1 _ CASE userflags[ n _ guindex()] OF
               < 0: 0;
               = 0: 3;
               ENDCASE 6;
            var2 _ CASE userflags[ m _ guindex()] OF
               < 0: 0;
               = 0: 1;
               ENDCASE 2;
            userflags[ n] _ ortable[ var1 + var2];
            END;
         RETURN;
         END;
      IF FIND *fand* $NP THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            var1 _ CASE userflags[ n _ guindex()] OF
               < 0: 0;
               = 0: 3;
               ENDCASE 6;
            var2 _ CASE userflags[ m _ guindex()] OF
               < 0: 0;
               = 0: 1;
               ENDCASE 2;
            userflags[ n] _ andtable[ var1 + var2];
            END;
         RETURN;
         END;
      IF FIND *fxor* $NP THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            var1 _ CASE userflags[ n _ guindex()] OF
               < 0: 0;
               = 0: 3;
               ENDCASE 6;
            var2 _ CASE userflags[ m _ guindex()] OF
               < 0: 0;
               = 0: 1;
               ENDCASE 2;
            userflags[ n] _ xortable[ var1 + var2];
            END;
         RETURN;
         END;
      IF FIND *fget* $NP THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            n _ guindex();
            FIND ^tp1;
            m _ guindex();
            userflags[ n] _
               IF m # 0 THEN userflags[ m]
               ELSE getfnum( FIND tp1 >);
            END;
```

```
      RETURN;
      END;
   IF FIND *fnot* $NP THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         n _ guindex();
         userflags[ n] _ - userflags[ n];
         END;
      RETURN;
      END;
   IF FIND *fsetf* $NP THEN
      BEGIN
      IF invert >= 0 THEN userflags[ guindex()] _ -1;
      RETURN;
      END;
   IF FIND *fsett* $NP THEN
      BEGIN
      IF invert >= 0 THEN userflags[ guindex()] _ 1;
      RETURN;
      END;
   IF FIND *fsetm* $NP THEN
      BEGIN
      IF invert >= 0 THEN userflags[ guindex()] _ 0;
      RETURN;
      END;
   IF FIND *fadd* $NP THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         var1 _ userflags[ n _ guindex()];
         FIND ^tp1;
         IF (m _ guindex()) # 0 THEN var2 _ userflags[ m]
         ELSE var2 _ getfnum( FIND tp1 >);
         userflags[ n] _ var1 + var2;
         END;
      RETURN;
      END;
   IF FIND *fsub* $NP THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         var1 _ userflags[ n _ guindex()];
         FIND ^tp1;
         IF (m _ guindex()) # 0 THEN var2 _ userflags[ m]
         ELSE var2 _ getfnum( FIND tp1 >);
         userflags[ n] _ var1 - var2;
         END;
      RETURN;
      END;
   IF FIND *csrept* THEN
      BEGIN
      grepeat _ invert;
      RETURN;
      END;
   IF FIND *csignr* THEN
```

```
        BEGIN
        IF invert >= 0 THEN gignore _ TRUE ELSE gignore _ -1;
        RETURN;
        END;
    IF FIND *csabrt* THEN
        BEGIN
        IF invert >= 0 THEN gabort _ inptrf _ TRUE
        ELSE gabort _ (inptrf _ 0) - 1;
        RETURN;
        END;
    IF FIND *cslogo* THEN
        BEGIN
        glogout _ invert;
        RETURN;
        END;
        %  wait not implemented in NLS 10
    wfor _ wuntil _ FALSE;
    IF FIND *cswait* $NP ("for" FS wfor / "until" FS wuntil) $NP
    ^tp1 1$D ^tp2 THEN
        IF invert >= 0 THEN
            BEGIN
            *12str* _ tp1 tp2;
            hours _ VALUE( $12str);
            IF FIND ": ^tp1 2D ^tp2 THEN
                BEGIN
                *12str* _ tp1 tp2;
                minutes _ VALUE( $12str);
                END
            ELSE minutes _ hours := 0;
            gwait _ ((hours * 60) + minutes) * 60;
            IF wfor THEN RETURN;
            IF (hours NOT IN [0, 23]) OR (minutes NOT IN [0, 59])
            THEN
                BEGIN
                gwait _ 0;
                GOTO rerr;
                END;
            !odcnv(0, -1, 0, 0);
                seconds _ R4.RH;
            gwait _ gwait - seconds;
            IF gwait <= 0 THEN gwait _ gwait + (24*60*60);
            RETURN;
            END
        ELSE RETURN;
        %
    IF (n _ guindex()) # 0 THEN
        BEGIN
        userflags[n] _ invert;
        RETURN;
        END;
    (rerr):                                                  1C32AV
    *locstr* _ "Illegal command: ignored", CR, LF, "      ", cps
    cpe;
    dismes( 1, $locstr);
    RETURN;
    END.
```

%%

1C32B@

```
(rnfl) PROCEDURE % copy printing file to <*prtdir*> directory %
                                                                    1C33
    % FORMALS %
      ;
    % LOCALS %
      LOCAL ret;
      LOCAL ojfn; % jfn of temporary copy %
      LOCAL njfn; % jfn of printer copy %
      LOCAL fsize[2];
      LOCAL dirnum;
      LOCAL STRING dirstr[50];
      LOCAL STRING locstr[400];
    % see if output should be copied to printer directory %
      IF gcopyprinter < 0 THEN RETURN;
    % initialize %
      ret _ FALSE;
    % get jfn of printing file %
      IF NOT SKIP !gtjfn( 1B11 .V 1B6, 18M6 .V ($gpfilename + 1)
      ) THEN
        GOTO rnflret;
      ojfn _ R1;
    % get name of connected directory %
      !gjinf();
        dirnum _ R2;
      IF SKIP !dirst( 18M6 .V ($dirstr + 1), dirnum) THEN
        BEGIN
        dirstr.L _ dirstr.M;
        dirnum _ 0;
        WHILE *dirstr*[dirnum + 1] DO BUMP dirnum;
        dirstr.L _ dirnum;
        IF dirstr.L THEN *dirstr* _ "<, *dirstr*, ">;
        END;
    % see if anything printed first %
      !gtfdb( ojfn, 2B6 .V 11B, $fsize);
      IF (fsize.RH = 0) AND (fsize[1] = 0) THEN
        BEGIN     % nothing printed %
        *locstr* _
          "No printing output - ", *dirstr*, *gpfilename*,  " :
           ";
        IF SKIP !delf( ojfn) THEN ret _ TRUE;
        IF ret THEN *locstr* _ *locstr*, "Deleted"
        ELSE *locstr* _ *locstr*, "UNDELETEABLE *****";
        GOTO rnfldis;
        END;
    % get jfn fordestination file %
      *locstr* _
        "<, *prtdir*, ">, *initsr*, "-, *gpfilename*,
        ";P777777", 0;
      IF NOT SKIP !gtjfn( 4B11 .V 1B6, 18M6 .V ($locstr + 1) )
      THEN
        BEGIN
        IF NOT SKIP !rljfn( ojfn) THEN NULL;
        GOTO rnflret;
        END;
      njfn _ R1;
    % copy and cleanup %
```

```
      ret _ cflapc( ojfn, njfn);
      IF ret THEN IF NOT SKIP !delnf( ojfn, 2) THEN NULL;
      IF NOT SKIP !rljfn( ojfn) THEN NULL;
      IF NOT SKIP !rljfn( njfn) THEN NULL;
   (rnflret):                                                    1C33J
      IF ret THEN
         *locstr* _ *dirstr*, *gpfilename*, " copied to <",
         *prtdir*, ">
      ELSE
         *locstr* _ "UNABLE to copy ", *dirstr*, *gpfilename*, "
         to <", *prtdir*, ">;
   (rnfldis):                                                    1C33K
      dismes( 1, $locstr);
   RETURN( ret);
   END.
   %%                                                            1C33N
```

```
(rplpdate) PROCEDURE % replace processing signature in passed
stid %                                                          1C34
   % FORMALS %
      ( flstid,    % statement stid in which to replace processing
      date %
      astr);       % adr string to insert before signature %
   REF astr;
   % LOCALS %
      LOCAL i;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING dtstr[50];
   tp1 _ flstid;   tp1[1] _ 1;
   LOOP
      IF NOT FIND tp1 > [")("] ^tp1 [")"] < 2CH ^tp2 THEN
         BEGIN
         ST tp1 _ SF(tp1) SE(tp1), TAB, "(())";
         REPEAT LOOP;
         END
      ELSE EXIT LOOP;
   !odtim( 18M6 .V ($dtstr + 1), -1, 0);
      dtstr.L _ dtstr.M;
      i _ 1;
      WHILE *dtstr*[i] DO BUMP i;
      dtstr.L _ i - 1;
   ST tp1 tp2 _ SP, *astr*, SP, *initsr*, ": ", *dtstr*, SP;
   RETURN;
   END.
   %%                                                           1C34J
```

```
(runfork) PROCEDURE % process one inferior fork processor %   1C35
   % FORMALS %
      (cp,          % char count in cfstid after *runfil*/*newproc*
      %
      spec);        % TRUE > get result from R1 at fork completion
      %
   % LOCALS %
      LOCAL fjfn;
      LOCAL ijfn;
      LOCAL forknum;
      LOCAL result;
      LOCAL savnm;
      LOCAL savrm;
      LOCAL savgoutput;
      LOCAL savpgoutput;
      LOCAL dtc;
      LOCAL TEXT POINTER tp1, tp2, rf1;
      LOCAL regs[20];
      LOCAL parray[50];
      LOCAL STRING locstr[2000];
      LOCAL STRING l3str[300];
      LOCAL STRING infname[200];
      LOCAL STRING inpname[200];
      LOCAL STRING stacop[2000];
   % get copy of driving statement for later copying %
      tp1 _ cfstid;   tp1[1] _ 1;
      *stacop* _ tp1 SE( tp1);
      cqplex();
   % initialize builtin flags %
      % indicate not run yet %
         infrun _ -1;   infnrun _ 1;
      % indicate run successfully %
         infok _ 1;   infnok _ -1;
   % tell user where we are %
      INVOKE (catrunfork1, retrunfork);
      rf1 _ cfstid;   rf1[1] _ cp;
      lnkprs( $rf1, $parray);
      lnbfls( 0, $parray, $infname);
      rf1[1] _ parray[le+1];
      lnbfls( $rf1, 0, $inpname);
      *locstr* _
         "Fork: ", *infname*, "  Input: ", *inpname*, "
         Processing started";
      dismes( 1, $locstr);
      DISABLE (catrunfork1);  %dropped at RETURN%
   % parse any pre commands (and handle output redirection) %
      savgoutput _ goutput;
      IF NOT preparsecmd( cfstid, cp : cp ) THEN
         BEGIN
         dismes( 1, $"Unable to parse pre commands for this
         statement, branch ignored");
         RETURN;
         END;
      IF gabort >= 0 THEN RETURN;
      IF gignore >= 0 THEN
         BEGIN
```

```
        dismes( 1, $"Branch ignored");
        RETURN;
        END;
     IF savgoutput # goutput THEN
        BEGIN
        jfntostr( goutput.RH, $13str, 011110B6 .V 1);
        *13str* _ "Changing primary output file to: ", *13str*;
        dismes( 1, $13str);
        !spjfn( 4000008, goutput);
        END;
   % check to see if we should run %
     IF grunfile < 0 THEN GOTO rfpost;
   % get jfn for inferior fork %
     fjfn _ ijfn _ forknum _ 0;
     INVOKE (catrunfork2, rfcleanup);
     *infname* _ *infname*, 0;
     IF NOT SKIP !gtjfn( 1B11.V 1B6, 18M6 .V ($infname + 1))
     THEN
        BEGIN
        dismes( 1, $"Unable to GTJFN for inferior fork");
        result _ -1;
        GOTO rfcleanup;
        END;
     fjfn _ R1;
   % get jfn for input file %
     *inpname* _ *inpname*, 0;
     IF NOT SKIP !gtjfn( 1B11.V 1B6, 18M6 .V ($inpname + 1))
     THEN
        BEGIN
        dismes( 1, $"Unable to GTJFN for inferior fork input
        file");
        result _ -1;
        GOTO rfcleanup;
        END;
     ijfn _ R1;
     IF NOT SKIP !openf( ijfn, 07B10 .V 2B5) THEN
        BEGIN
        dismes( 1, $"Unable to OPENF for inferior fork input
        file");
        result _ -1;
        GOTO rfcleanup;
        END;
   % save our own state %
     !getnm();
        savnm _ R1;
     !gjinf();
        dtc _ R4;
     IF dtc # -1 THEN
        BEGIN
        !rfmod( 100B);
           savrm _ R2;
        END;
   % create and get into inferior fork %
     IF NOT SKIP !cfork( 2B11) THEN
        BEGIN
        dismes( 1, $"Unable to OPENF for inferior fork input
```

```
            file");
            result _ -1;
            GOTO rfcleanup;
            END;
         forknum _ R1;
         !epcap( forknum, -1, -1);
         !spjfn( forknum, (ijfn * 1B6) .V goutput.RH);
         !stiw( forknum, 0);
         R1.LH _ forknum;   R1.RH _ fjfn;
            !get();
         IF NOT SKIP !rljfn( fjfn := 0) THEN NULL;
      % run it, wait, and get result %
         dismes( 1, $"Starting execution");
         infrun _ 1;   infnrun _ -1;
         BUMP numinferiors;
         !sfrkv( forknum, 0);
         !wfork();
         !rfsts( forknum);
            R1 _ R1 .A 7M6;   R1 _ R1.LH;
            IF R1 # 2 THEN result _ -1
            ELSE
               IF NOT spec THEN result _ 1
               ELSE
                  BEGIN
                  !rfacs( forknum, $regs);
                  result _ regs[1];
                  END;
      (rfcleanup): % cleanup %                                        1C35M
         DROP(catrunfork2);
         dismes( 1, $"Execution completed");
         IF forknum AND ijfn THEN !spjfn( forknum, goutput);
         IF forknum THEN !kfork( forknum := 0);
         IF fjfn THEN IF NOT SKIP !closf( fjfn := 0) THEN NULL;
         IF ijfn THEN IF NOT SKIP !closf( ijfn := 0) THEN NULL;
         !setnm( savrm);
         IF dtc # -1 THEN
            BEGIN
            !sfmod( 100B, savrm);
            !stpar( 100B, savrm);
            END;
         infok _ result;   infnok _ - infok;
         IF result >= 0 THEN BUMP numokinferiors;
      (rfpost): % process any post commands %                         1C35N
         savpgoutput _ goutput;
         IF NOT postparsecmd( cfstid, cp ) THEN
            BEGIN
            dismes( 1, $"Unable to parse post commands for this
            branch");
            RETURN;
            END;
         IF savpgoutput # goutput THEN
            BEGIN
            jfntostr( goutput.RH, $13str, 01111086 .V 1);
            *13str* _ "Changing primary output file to: ", *13str*;
            dismes( 1, $13str);
            !spjfn( 400000B, goutput);
```

```
        IF savpgoutput # savgoutput THEN clspjfn(
        savpgoutput.RH);
        END;
     IF gabort >= 0 THEN RETURN;
  % indicate branch processed %
     tp1 _ tp2 _ $stacop;
        tp1.stastr _ tp2.stastr _ TRUE;
        tp1[1] _ 1;
        tp2[1] _ stacop.L + 1;
        cinssta( procstid, -1, $tp1, $tp2);
     IF infrun = 1 THEN
        IF infok >= 0 THEN *locstr* _ "Processed and Run
        Successfully by"
        ELSE *locstr* _ "Processed and Run UNSUCCESSFULLY by"
     ELSE *locstr* _ "Processed by";
     rplpdate( cfstid, $locstr);
     IF infrun = 1 THEN  ccopsta( lhsstid, -1, cfstid, 0, 0);
  dismes( 1, $"Done processing");
  (retrunfork):                                                  1C35Q
  DROP (catrunfork1);  %other catchprase DROP'd earlier%
  RETURN;
  % catchphrases %
     (catrunfork1) CATCHPHRASE();                                1C35T1
        BEGIN
        CASE SIGNALTYPE OF
           = notetype : NULL;
           = helptype : NULL;
           = aborttype :
              BEGIN
              DISABLE (catrunfork1);
              dismes( 1, $"Unable to parse inferior fork
              statement; branch ignored");
              result _ -1;
              TERMINATE;  %goes to 'retrunfork'%;
              END;
           ENDCASE;
        CONTINUE;
        END;
     (catrunfork2) CATCHPHRASE();                                1C35T2
        BEGIN
        CASE SIGNALTYPE OF
           = notetype : NULL;
           = helptype : NULL;
           = aborttype :
              BEGIN
              DISABLE (catrunfork2);
              dismes( 1, $"Unable to run inferior fork");
              TERMINATE;  %goes to 'rfcleanup' %
              END;
           ENDCASE;
        CONTINUE;
        END;
  END.
  %%                                                             1C35V
```

```
(upconcnt) PROCEDURE % update condtional count driving statement
%                                                                  1C36
   % FORMALS %
     ;
   % LOCALS %
     LOCAL TEXT POINTER tp1, tp2;
   ccopsta( constid, -1, constid, FALSE, FALSE);
   tp1 _ constid;   tp1[1] _ 1;
   IF NOT FIND tp1 > $NP $D $NP ^tp2 "Condtionals" THEN
      ST tp1 _ STRING(condcnt), " Conditionals;   (())"
   ELSE ST tp1 tp2 _ STRING( condcnt), SP;
   rplpdate( constid, $"Counted by");
   RETURN;
   END.
   %%                                                              1C36J
```

```
(upfile) PROCEDURE % update file according to gupdate %          1C37
   % FORMALS %
     ;
   % LOCALS %
     LOCAL fl;   REF fl;
     LOCAL type;
     LOCAL fileno;
     LOCAL STRING flname[200];
     LOCAL STRING locstr[300];
   % make sure really needs update %
     &fl _ flntadr( fileno _ srcstid.stfile);
     IF (NOT fl.flpart) AND (guptype # 3) THEN RETURN( 0);
     IF fl.fllock AND (NOT fl.flpart) THEN
        BEGIN
        wrtmsg( 0, $": File cannot be updated - locked by
        someone else");
        RETURN( 0);
        END;
   % tell user which file we are updateing %
     *locstr* _ ": Updating (", *[( CASE guptype OF
        = 1: $"old";
        = 2: $"new";
        = 3: $"compact";
        ENDCASE $"new" )]*, ");
     wrtmsg( 0, $locstr);
   % do it %
     type _ CASE guptype OF
        = 1: oldversion;
        = 2: newversion;
        = 3: upcompact;
        ENDCASE newversion;
     IF type = upcompact THEN clist( 15, fileno, 0);
     cupdfil( fileno, type, 0);
   % tell user we are done %
     BUMP numupdates;
     ccopsta( uphstid, -1, uphstid, FALSE, FALSE);
     rplpdate( uphstid, $"Updated by");
     *locstr* _ ": Done ", *locstr*[3 TO locstr.L];;
     wrtmsg( 0, $locstr);
   RETURN( 1);
   END.
   %%                                                            1C37I
```

```
(upinccnt) PROCEDURE % update include count driving statement %
                                                                    1C38
   % FORMALS %
     ;
   % LOCALS %
     LOCAL TEXT POINTER tp1, tp2;
   ccopsta( incstid, -1, incstid, FALSE, FALSE);
   tp1 _ incstid;   tp1[1] _ 1;
   IF NOT FIND tp1 > $NP $D $NP ^tp2 "Includes" THEN
     ST tp1 _ STRING(incldcnt), " Includes;      (())"
   ELSE ST tp1 tp2 _ STRING( incldcnt), SP;
   rplpdate( incstid, $"Counted by");
   RETURN;
   END.
   %%                                                               1C38J
```

```
(upsysgd) PROCEDURE % update a sysgd file %                          1C39
   % FORMALS %
      ( flstid);  % origin stid of sysgd file %                     1C39A1
   % LOCALS %
      (da) REF;                                                     1C39B1
      LOCAL fl;   REF fl;
      LOCAL plsstid;
      LOCAL plestid;
   &da _ lda();  %needed by 'sortgrp'%
   &fl _ flntadr( flstid.stfile);
   IF fl.flpart THEN
      BEGIN
      wrtmsg( -flstid.stfile, $": Updating (Compact)");
      IF (plsstid _ getnxt( flstid)) # endfil THEN
         IF (plestid _ getail( plsstid)) # plsstid THEN
            sortgrp( plsstid, plestid, da.dawid);
      clist( 15, flstid.stfile, 0);
      cupdfil( flstid.stfile, upcompact, 0 );
      wrtmsg( -flstid.stfile, $": Done Updating (Compact)");
      END;
   RETURN;
   END.
   %%                                                               1C39B
```

```
(wrtmsg) PROCEDURE % display message on output file %          1C40
   % FORMALS %
      ( flstid,    % 0 / statement stid containing file name / -
      file number %
      astr);       % adr message %
   REF astr;
   % LOCALS %
      LOCAL TEXT POINTER tp1, tp2, tp3, tp4;
      LOCAL parray[40];
      LOCAL STRING flname[200];
      LOCAL STRING locstr[200];
   CASE tp1 _ flstid OF
      > 0:
         BEGIN
         tp1[1] _ 1;
         lnkprs( $tp1, $parray);
         tp1 _ parray[ls];    tp1[1] _ parray[ls+1];
         tp2 _ parray[le];    tp2[1] _ parray[le+1];
         END;
      = 0:
         BEGIN
         tp1 _ garray[ls];    tp1[1] _ garray[ls+1];
         tp2 _ garray[le];    tp2[1] _ garray[le+1];
         END;
      < 0:
         BEGIN
         filnam( -flstid, $flname);
         tp1 _ tp2 _ $flname;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ flname.L + 1;
         IF FIND tp1 > [";] < CH ^tp3 > [",] ^tp4 THEN
            BEGIN
            *flname* _ tp1 tp3, ",, tp4 tp2;
            tp2[1] _ flname.L + 1;
            END;
         END;
      ENDCASE;
   *locstr* _ - tp1 tp2, *astr*;
   dismes( 1, $locstr);
   RETURN;
   END.
   %%                                                          1C40I
```

```
(nnumon) PROCEDURE (sw );                                              1C41
   LOCAL i, stid, stnv[100];
   LOCAL TEXT POINTER tp1;
   REF sw;
   IF (fchtxt( stid _ sw.swcstid) > 1) OR (inrecord AND FIND
   ["[])
   OR FIND "%%" / ( "% ("+/"-) UL $ULD "% ) THEN
```

```
        BEGIN
        IF NOT (sw.swvspec.vsstnf := TRUE) THEN stvect( stid,
        sw.swsvw);
        sw.swvspec.vsstnr _ TRUE;
        sw.swvspec.vssidf _ FALSE;
        END
    ELSE sw.swvspec.vsstnf _ FALSE;
    RETURN;
    END.
    %%                                                    1C41H
```

```
(finder) PROCEDURE  % content analyzer %                                1C42
    % FORMALS %
      (sw);                                                             1C42A1
    LOCAL
      frtchr,
      da,
      stid,
      i,
      clvl,
      char,
      cmflg;
    LOCAL
      spnam, splnk, spnum;
    LOCAL TEXT POINTER
      tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8;
    REF
      da, sw;
    spnam _ 15;
    splnk _ 49;
    spnum _ 70;
    % take care of getting numbers on first %
      FIND ^tp1;
      nnumon( &sw);
      FIND tp1 >;
    IF NOT infile THEN
      BEGIN
      tp1[1] _ fchtxt( sw.swcstid);
      IF FIND tp1 > $NP "FILE" THEN infile _ TRUE;
      RETURN( TRUE);
      END;
    IF gconcnt >= 0 THEN
      IF FIND tp1 > '% ('+/'-) UL $ULD '% THEN BUMP condcnt;
    IF ginccnt >= 0 THEN
      IF FIND tp1 > "INCLUDE" 1$NP THEN BUMP incldcnt;
    IF gsysgd < 0 THEN RETURN( TRUE);
    IF inproc THEN
      BEGIN
      IF FIND tp1 > "END" '. THEN inproc _ FALSE;
      RETURN( TRUE);
      END;
    IF inrecord THEN
      BEGIN
      % get tptrs around name %
        IF NOT FIND tp1 ^tp5 ^tp6 > [''] ^tp7 < CH $NP ^tp2 > CH
        THEN
          RETURN( TRUE);
      cmflg _ FALSE;
      LOOP
        IF FIND ['% / ';] < CH ^tp5 > THEN
          BEGIN
          CASE READC OF
            = '%:
              BEGIN
              FIND ['% / ENDCHR] ^tp6;
              cmflg _ TRUE;
              END;
```

```
                = ";:
                   BEGIN
                   inrecord _ FALSE;
                   EXIT LOOP;
                   END;
                ENDCASE;
             END
          ELSE EXIT LOOP;
       % now get any fdrcomment %
          IF NOT cmflg THEN
             IF FIND tp7 > [";] THEN
                BEGIN
                FIND < CH ^tp5 > CH [";/ ENDCHR] ^tp6;
                END;
       % get field fdrwidth %
          IF NOT FIND tp7 > $NP ^tp7 1$ULD ^tp8 $NP "] THEN FIND
          tp7 ^tp8;
          *fdrwidth* _ - tp7 tp8;
          IF NOT fdrwidth.L THEN *fdrwidth* _ "subfields";
       % build string to insert %
          *fdrstnum* _ NULL;
             fechnm( sw.swsvw, $fdrstnum);
          *fdrstmnt* _ "(, tp1 tp2, ");
             DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
             spnam;
             *fdrstmnt* _ *fdrstmnt*, *fname*, " 0", STRING(
             getsid( tp1)), ">;
             DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
             splnk;
             *fdrstmnt* _ *fdrstmnt*, "FIELD - ", *fdrwidth*;
             DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
             spnum;
             *fdrstmnt* _ *fdrstmnt*, *fdrstnum*;
             IF tp5[1] < tp6[1] THEN *fdrstmnt* _ *fdrstmnt*, CR,
             "   ", tp5 tp6;
       % insert statement in sysgd file %
          tp1 _ tp2 _ $fdrstmnt;
             tp1.stastr _ tp2.stastr _ TRUE;
             tp1[1] _ 1;
             tp2[1] _ fdrstmnt.L + 1;
          cinssta( sysorg, -1, $tp1, $tp2);
       RETURN( TRUE);
       END;
    frtchr _ fchtxt( sw.swcstid);
    IF (frtchr > 1) AND FIND tp1 > ["PROCEDURE"/"COROUTINE"] ^tp6
    < $UL ^tp5 THEN LOOP
       BEGIN
       FIND $NP;
       CASE char _ READC OF
          = "):
             IF NOT FIND $NP ^tp2 $-"( >$NP ^tp1 THEN EXIT LOOP;
          = "%:
             BEGIN
             FIND ["%] $NP;
             REPEAT CASE;
             END;
```

```
        ENDCASE EXIT LOOP;
    inproc _ TRUE;
    % tp5 & tp6 surround PROCEDURE/COROUTINE %
    % tp1 & tp2 surround routine name %
    % now collect comments and formals %
        *fdrcomment* _ NULL; cmflg _ FALSE;
        *fdrparams* _ NULL;
        FIND tp6 ^tp3 >;
        CASE READC OF
            = ENDCHR:
                BEGIN
                IF (tp3 _ getnxt( tp3) ) = endfil THEN EXIT LOOP;
                tp3[1] _ 1;
                FIND tp3 >;
                REPEAT CASE;
                END;
            = NP: REPEAT CASE;
            = ":
                IF NOT cmflg THEN
                    IF FIND [".] < CH ^tp3 > CH [".  /ENDCHR] ^tp4
                    THEN
                        *fdrcomment* _ tp3 tp4
                    ELSE
                        IF FIND tp2 > [".] < CH ^tp3 > CH [".
                        /ENDCHR] ^tp4 THEN
                            *fdrcomment* _ tp3 tp4;
            = "%:
                BEGIN
                IF NOT cmflg THEN *fdrcomment* _ "%;
                CASE char _ READC OF
                    = "%: IF NOT cmflg THEN *fdrcomment* _
                    *fdrcomment*, "%;
                    = ENDCHR:
                        BEGIN
                        IF NOT cmflg THEN *fdrcomment* _
                        *fdrcomment*, SP;
                        IF (tp3 _ getnxt(tp3)) = endfil THEN EXIT
                        LOOP;
                        tp3[1] _ 1;
                        FIND tp3 >;
                        REPEAT CASE;
                        END;
                    ENDCASE
                        BEGIN
                        IF NOT cmflg THEN *fdrcomment* _
                        *fdrcomment*, char;
                        REPEAT CASE;
                        END;
                cmflg _ TRUE;
                REPEAT CASE;
                END;
            = "(:
                BEGIN
                CASE char _ READC OF
                    = "): REPEAT CASE 2;
                    = "%:
```

```
                    CASE char _ READC OF
                       = '%: REPEAT CASE 2;
                       = ENDCHR:
                          BEGIN
                          IF (tp3 _ getnxt(tp3)) = endfil THEN
                          EXIT LOOP;
                          tp3[1] _ 1;
                          FIND tp3 >;
                          REPEAT CASE;
                          END;
                       ENDCASE REPEAT CASE;
                 = LD:
                    BEGIN
                    IF fdrparams.L THEN *fdrparams* _
                    *fdrparams*, ", ";
                    *fdrparams* _ *fdrparams*, char;
                    CASE char _ READC OF
                       = ',:
                          BEGIN
                          FIND ^tp3;
                          IF FIND SE(*fdrparams*) < $NP ^tp4
                          THEN
                             fdrparams.L _ tp4[1] - 1;
                          FIND tp3 >;
                          REPEAT CASE 2;
                          END;
                       = '):
                          BEGIN
                          FIND ^tp3;
                          IF FIND SE(*fdrparams*) < $NP ^tp4
                          THEN
                             fdrparams.L _ tp4[1] - 1;
                          FIND tp3 >;
                          REPEAT CASE 2 (char);
                          END;
                       = '%:
                          CASE char _ READC OF
                             = '%: REPEAT CASE 2;
                             = ENDCHR:
                                BEGIN
                                IF (tp3 _ getnxt(tp3)) = endfil
                                THEN
                                   EXIT LOOP;
                                tp3[1] _ 1;
                                FIND tp3 >;
                                REPEAT CASE;
                                END;
                             ENDCASE REPEAT CASE;
                       = ENDCHR:
                          BEGIN
                          *fdrparams* _ *fdrparams*, SP;
                          IF (tp3 _ getnxt(tp3)) = endfil THEN
                          EXIT LOOP;
                          tp3[1] _ 1;
                          FIND tp3 >;
                          REPEAT CASE;
```

```
                        END;
                     ENDCASE
                        BEGIN
                        *fdrparams* _ *fdrparams*, char;
                        REPEAT CASE;
                        END;
                  REPEAT CASE;
                  END;
               = ENDCHR:
                  BEGIN
                  IF (tp3 _ getnxt(tp3)) = endfil THEN EXIT
                  LOOP;
                  tp3[1] _ 1;
                  FIND tp3 >;
                  REPEAT CASE;
                  END;
               ENDCASE REPEAT CASE;
            REPEAT CASE;
            END;
         ENDCASE;
      % build string to insert %
         *fdrstnum* _ NULL;
            fechnm( sw.swsvw, $fdrstnum);
         *fdrstmnt* _ "(, tp1 tp2, ");
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            spnam;
            *fdrstmnt* _ *fdrstmnt*, *fname*, " 0", STRING(
            getsid( tp1)), ">;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            splnk;
            *fdrstmnt* _ *fdrstmnt*, tp5 tp6;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            spnum;
            *fdrstmnt* _ *fdrstmnt*, *fdrstnum*;
            IF *fdrcomment* = "% FORMALS %" THEN *fdrcomment* _
            NULL;
            IF fdrparams.L THEN
               *fdrstmnt* _ *fdrstmnt*, CR, "   FORMAL PARAMTERS(
               ", *fdrparams*, ")";
            IF fdrcomment.L THEN *fdrstmnt* _ *fdrstmnt*, CR, "
            ", *fdrcomment*;
      % insert statement in sysgd file %
         tp1 _ tp2 _ $fdrstmnt;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ fdrstmnt.L + 1;
         cinssta( sysorg, -1, $tp1, $tp2);
      RETURN( TRUE);
      END;
   IF (tp1[1] _ frtchr) > 1 THEN LOOP
      BEGIN
      % get tptrs around name %
         IF NOT FIND tp1 ^tp3 < ") $NP ^tp2 $-"( > $NP ^tp1 THEN
         EXIT LOOP;
      % get fdrtype of declaration %
         FIND tp3 > $NP;
```

```
        CASE READC OF
           = ";:
              BEGIN
              *fdrtype* _ "LOCAL";
              FIND ^tp6;
              END;
           = UL:
              BEGIN
              FIND < CH ^tp5  > $UL ^tp6 ^tp8 $NP;
              *fdrtype* _ tp5 tp6;
              CASE READC OF
                 = UL:
                    BEGIN
                    FIND < CH ^tp5  > $UL ^tp6 ^tp8 $NP;
                    *fdrtype* _ *fdrtype*, SP, tp5 tp6;
                    REPEAT CASE;
                    END;
                 = "%:
                    BEGIN
                    FIND [^% / ENDCHR] $NP;
                    REPEAT CASE;
                    END;
                 ENDCASE;
              FIND tp6 > [^; / ENDCHR] ^tp6;
              END;
           = "%:
              BEGIN
              FIND [^% / ENDCHR] $NP;
              REPEAT CASE;
              END;
           ENDCASE
              BEGIN
              *fdrtype* _ "LOCAL";
              FIND ^tp6;
              END;
        IF *fdrtype* = "RECORD" THEN inrecord _ TRUE
        ELSE
           BEGIN
           IF FIND SF(*fdrtype*) "EXTERNAL" $NP ^tp5 THEN
              *fdrtype* _ "EXT ", tp5 SE(*fdrtype*);
           IF FIND SF(*fdrtype*) [^"CONSTANT" / "ADDRESS"] THEN
              IF FIND tp8 > $NP "= $NP ^tp5 [^;] < CH $NP ^tp7
              THEN
                 *fdrtype* _ *fdrtype*, " =", tp5 tp7;
           END;
     % now get any fdrcomment %
        FIND tp6 > ^tp5;
        IF FIND [^%] THEN FIND < CH ^tp5 > CH [^% / ENDCHR]
        ^tp6;
     % build string to insert %
        *fdrstnum* _ NULL;
           fechnm( sw.swsvw, $fdrstnum);
        *fdrstmnt* _ "(, tp1 tp2, ");
           DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
           spnam;
           *fdrstmnt* _ *fdrstmnt*, *fname*, " 0", STRING(
```

```
            getsid( tp1)), ">;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            splnk;
            *fdrstmnt* _ *fdrstmnt*, *fdrtype*;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            spnum;
            *fdrstmnt* _ *fdrstmnt*, *fdrstnum*;
            IF tp5[1] < tp6[1] THEN *fdrstmnt* _ *fdrstmnt*, CR,
            "   ", tp5 tp6;
      % insert statement in sysgd file %
         tp1 _ tp2 _ $fdrstmnt;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ fdrstmnt.L + 1;
         cinssta( sysorg, -1, $tp1, $tp2);
      RETURN( TRUE);
      END;
   tp1[1] _ 1;
   IF FIND tp1 > "FINISH" THEN infile _ FALSE;
   RETURN( TRUE );
   END.
   %%                                                    1C42W
```

```
% REMOVE THIS WHEN NEW NLS WITH CORRECTED "cnvvsp" HAS BEEN MADE
%
   (cnvvsp) %convert viewspec string to viewspec words%
   PROCEDURE( vstring REF %viewspec string%, windowid);        1C43A
      %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 _ aloblk(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
%%                                                          1C45
```

(aarray)	<nine, library, 06322>	EXT	1B2O
(always)	<nine, library, 06364>	EXT STRING	1B2AA3
(andtable)	<nine, library, 06333>	EXT	1B2A@2
(anyaction)	<nine, library, 06489>	EXT	1B2AD2
(argxcw)	<nine, library, 010354>	EXT CONSTANT =1	1B2AE3
(argxvs)	<nine, library, 010362>	EXT CONSTANT =2	1B2AE4
(bvar1)	<nine, library, 06401>	EXT STRING	1B2AC4
(bvar10)	<nine, library, 06410>	EXT STRING	1B2AC1
(bvar11)	<nine, library, 06411>	EXT STRING	1B2AC1
(bvar12)	<nine, library, 06412>	EXT STRING	1B2AC1
(bvar13)	<nine, library, 06413>	EXT STRING	1B2AC1
(bvar14)	<nine, library, 06414>	EXT STRING	1B2AC1
(bvar15)	<nine, library, 06415>	EXT STRING	1B2AC1
(bvar16)	<nine, library, 06416>	EXT STRING	1B2AC1
(bvar17)	<nine, library, 06417>	EXT STRING	1B2AC2
(bvar18)	<nine, library, 06418>	EXT STRING	1B2AC2
(bvar19)	<nine, library, 06419>	EXT STRING	1B2AC2
(bvar2)	<nine, library, 06402>	EXT STRING	1B2AC5
(bvar20)	<nine, library, 06420>	EXT STRING	1B2AC2
(bvar21)	<nine, library, 06421>	EXT STRING	1B2AC2
(bvar22)	<nine, library, 06422>	EXT STRING	1B2AC2
(bvar23)	<nine, library, 06423>	EXT STRING	1B2AC2
(bvar24)	<nine, library, 06424>	EXT STRING	1B2AC2
(bvar25)	<nine, library, 06425>	EXT STRING	1B2AC2
(bvar26)	<nine, library, 06426>	EXT STRING	1B2AC2
(bvar27)	<nine, library, 06427>	EXT STRING	1B2AC3
(bvar28)	<nine, library, 06428>	EXT STRING	1B2AC3
(bvar29)	<nine, library, 06429>	EXT STRING	1B2AC3
(bvar3)	<nine, library, 06403>	EXT STRING	1B2AC6
(bvar30)	<nine, library, 06430>	EXT STRING	1B2AC3
(bvar31)	<nine, library, 06431>	EXT STRING	1B2AC3
(bvar32)	<nine, library, 06432>	EXT STRING	1B2AC3
(bvar33)	<nine, library, 06433>	EXT STRING	1B2AC3
(bvar34)	<nine, library, 06434>	EXT STRING	1B2AC3
(bvar35)	<nine, library, 06435>	EXT STRING	1B2AC3
(bvar36)	<nine, library, 06436>	EXT STRING	1B2AC3
(bvar37)	<nine, library, 06437>	EXT STRING	1B2AC4
(bvar38)	<nine, library, 06438>	EXT STRING	1B2AC4
(bvar39)	<nine, library, 06439>	EXT STRING	1B2AC4
(bvar4)	<nine, library, 06404>	EXT STRING	1B2AC7
(bvar40)	<nine, library, 06440>	EXT STRING	1B2AC4
(bvar5)	<nine, library, 06405>	EXT STRING	1B2AC8
(bvar6)	<nine, library, 06406>	EXT STRING	1B2AC9
(bvar7)	<nine, library, 06407>	EXT STRING	1B2AC1
(bvar8)	<nine, library, 06408>	EXT STRING	1B2AC1
(bvar9)	<nine, library, 06409>	EXT STRING	1B2AC1
(bvarstr)	<nine, library, 06442>	EXT	1B2AC4
(carray)	<nine, library, 06323>	EXT	1B2P
(cfstid)	<nine, library, 06306>	EXT	1B2H5
(clsfile)	<nine, library, 07465>	PROCEDURE	1C5
(clspjtn)	<nine, library, 07476>	PROCEDURE	1C6
(cmdbra)	<nine, library, 06395>	EXT STRING	1B2AB5
(cmperr)	<nine, library, 06523>	EXT	1B2AD2
(cmpfile)	<nine, library, 07543>	PROCEDURE	1C7
(cmpok)	<nine, library, 06524>	EXT	1B2AD2
(cmpptr)	<nine, library, 09889>	EXT TEXT POINTER	1B2Q

```
(sdrive)        <nine, library, 06533>          EXT                     1B2AD7
(sourcedate)    <nine, library, 06500>          EXT                     1B2AD2
(sprstid)       <nine, library, 06303>          EXT                     1B2H2
(srcfil)        <nine, library, 06392>          EXT STRING              1B2AB2
(srcstid)       <nine, library, 06307>          EXT                     1B2H6
(srcwrd)        <nine, library, 06317>          EXT                     1B2J
(srtjfns)       <nine, library, 06320>          EXT                     1B2M
(supfile)       <nine, library, 09873>          EXT STRING              1B2Y
(sysorg)        <nine, library, 06302>          EXT                     1B2H1
(sysstid)       <nine, library, 06311>          EXT                     1B2H10
(tfe)           <nine, library, 06381>          EXT STRING              1B2AA4
(tff)           <nine, library, 06376>          EXT STRING              1B2AA4
(tfg)           <nine, library, 06383>          EXT STRING              1B2AA4
(tfi)           <nine, library, 06385>          EXT STRING              1B2AA5
(tfm)           <nine, library, 06377>          EXT STRING              1B2AA4
(tfne)          <nine, library, 06382>          EXT STRING              1B2AA4
(tfnf)          <nine, library, 06379>          EXT STRING              1B2AA4
(tfng)          <nine, library, 06384>          EXT STRING              1B2AA5
(tfnl)          <nine, library, 06386>          EXT STRING              1B2AA5
(tfnm)          <nine, library, 06380>          EXT STRING              1B2AA4
(tfnt)          <nine, library, 06378>          EXT STRING              1B2AA4
(tft)           <nine, library, 06375>          EXT STRING              1B2AA4
(topalmost)     <nine, library, 07054>          PROCEDURE               1C4
(topstid)       <nine, library, 06304>          EXT                     1B2H3
(totalerrors)   <nine, library, 06521>          EXT                     1B2AD2
(typwrtr)       <nine, library, 010357>         EXT CONSTANT =52        1B2AE1
(uflag)         <nine, library, 06400>          EXT STRING              1B2AC3
(upconcnt)      <nine, library, 09363>          PROCEDURE               1C36
(upfile)        <nine, library, 09377>          PROCEDURE               1C37
(uphstid)       <nine, library, 06309>          EXT                     1B2H8
(upinccnt)      <nine, library, 09418>          PROCEDURE               1C38
(upsysgd)       <nine, library, 09432>          PROCEDURE               1C39
(userflags)     <nine, library, 06529>          EXT                     1B2AD3
(writedate)     <nine, library, 06387>          EXT STRING              1B2AA5
(wrtmsg)        <nine, library, 09453>          PROCEDURE               1C40
(xlbevtop)      <nine, library, 06818>          PROCEDURE               1B3D
(xlchkadr)      <nine, library, 010577>         PROCEDURE               1B3E
(xldelhis)      <nine, library, 06864>          PROCEDURE               1B3F
(xlicompile)    <nine, library, 06990>          PROCEDURE               1B3G
(xliinit)       <nine, library, 07031>          PROCEDURE               1B3H
(xlreset)       <nine, library, 010494>         PROCEDURE               1B3I
(xortable)      <nine, library, 06334>          EXT                     1B2A@3
```

```
< NINE, LIBRARY.NLS;28, >, 27-May-78 21:14 BLP ;;;;
%source-code%                                                          1
   %grammar% FILE library  % (arcsubsys, cml10,)  (relnine,
   library.cgr,) %                                                     1A
      % COMPILE INSTRUCTIONS %
        INCLUDE <nine, nls-grammar, flags !subsystems>
      % DECLARATIONS %
        INCLUDE <nine, nls-grammar, declarations !universal>
        DECLARE COMMAND WORD
           "DISPLAY" = 51,
           "TYPEWRITER" = 52;
        DECLARE FUNCTION
           xlbevtop, xlchkadr, xldelhis, xlicompile, xliinit, xlreset,
           xsimulate;
        DECLARE VARIABLE namfil, diagnos;
      % COMMON RULES %
        INCLUDE <nine, nls-grammar, rules !universal>
      % COMMANDS % SUBSYSTEM library KEYWORD "LIBRARY"
        INITIALIZATION ziinit = xliinit();
        COMMAND  zpcompile = "COMPILE"    sent _ "FILE"
           <"at"> source _ DSEL(#"STATEMENT")
           <"using"> param _ LSEL(#"OLDFILENAME")
           <"to file"> namfil _ LSEL(#"NEWFILENAME")
           ( "RECORD" <"messages at"> diagnos _ DSEL(#"STATEMENT")
              CONFIRM
              xlicompile( sent, source, param, namfil, diagnos )
            / CONFIRM
              xlicompile( sent, source, param, namfil, NULL )
           );
        %COMMAND ziinclude = "INSTITUTE" <"include sequence
        generator">
           CONFIRM xliinstitute();%
        COMMAND zdelete="DELETE"
           <"history before date and time"> param _ LSEL(#"TEXT")
           <"in library form at"> source _ DSEL(#"BRANCH")
           CONFIRM
           xldelhis( param, source);
        COMMAND zprocess="PROCESS"
           <"library form at"> source _ DSEL(#"BRANCH")
           CONFIRM
           xlchkadr(source)
              %check for valid address before simulating a typewriter.
               Otherwise if a failure occurs during argument
              conversion, the command will be aborted by the CLI and
              the terminal will be left in typewriter mode.%
           ( IF DISPLAY  %simulate typewriter mode%
              xsimulate( #"TYPEWRITER" )
              simtty()
              ttysim _ TRUE
           / IF NOT DISPLAY
              ttysim _ FALSE )
           xlbevtop( source)  %do the Process command%
           ( IF ttysim  %go back to display mode%
              simdisplay()
              xlreset(#"DISPLAY", source)  %reset display%
              ttysim _ FALSE
```

```
      / IF NOT ttysim );
    INCLUDE <nine, nls-grammar, commands !universal>
    END.
FINISH
```

```
%be% FILE librarybe % using (arcsubsys, 1109,) to (relnine,
library.subsys,) %                                                    1B
   ALLOW!
   % Declarations %
      % Dispatch Table %
         (library) EXTERNAL _ (                                       1B2A1
            $"XLBEVTOP", $xlbevtop,
            $"XLCHKADR", $xlchkadr,
            $"XLDELHIS", $xldelhis,
            $"XLICOMPILE", $xlicompile,
            $"XLIINIT", $xliinit,
            $"XLRESET", $xlreset,
            0,0);
      (condcnt) EXTERNAL;     % conditional count while printing %
                                                                      1B2B
      (incldcnt) EXTERNAL;    % include count while printing %        1B2C
      (filenum) EXTERNAL;     % filenumber while printing %           1B2D
      (infile) EXTERNAL;      % indicates within code file while
      printing %                                                      1B2E
      (inrecord) EXTERNAL;    % indicates within record declaration
      while printing %                                                1B2F
      (inproc) EXTERNAL;      % indicates within procedure body while
      printing %                                                      1B2G
      % stids %
         (sysorg) EXTERNAL;   % stid index file origin %             1B2H1
         (sprstid) EXTERNAL;  % stid sysguide file origin %          1B2H2
         (topstid) EXTERNAL;  % stid library branch %                1B2H3
         (lhsstid) EXTERNAL;  % stid library history branch %        1B2H4
         (cfstid) EXTERNAL;   % stid current library branch statement
         %                                                            1B2H5
         (srcstid) EXTERNAL;  % stid current source file origin %
                                                                      1B2H6
         (procstid) EXTERNAL;         % stid current process hstry
         stmnt in library branch %                                    1B2H7
         (uphstid) EXTERNAL;  % stid current update history stmnt in
         library branch %                                             1B2H8
         (prntstid) EXTERNAL;         % stid current print history
         stmnt in library branch %                                    1B2H9
         (sysstid) EXTERNAL;  % stid current index stmnt in library
         branch %                                                    1B2H10
         (relstid) EXTERNAL;  % stid current compile stmnt in library
         branch %                                                    1B2H11
         (constid) EXTERNAL;  % stid current conditional stmnt in
         library branch %                                            1B2H12
         (incstid) EXTERNAL;  % stid current include stmnt in library
         branch %                                                    1B2H13
      (libsrttime) EXTERNAL;  % library processing start time %      1B2I
      (srcwrd) EXTERNAL;      % source file write date; 35M if file
      has PC %                                                        1B2J
      (reldate) EXTERNAL;     % rel file date %                       1B2K
      (newsysgdcnt) EXTERNAL;         % count of new sysgds generated
      %                                                               1B2L
      (srtjfns) EXTERNAL;     % starting primary jfns %               1B2M
      (garray) EXTERNAL [40];                                         1B2N
      (aarray) EXTERNAL [40]; % parsed link array for compile at
```

```
branch address %                                                    1B2O
(carray) EXTERNAL [40];  % parsed link array for compiler %
                                                                    1B2P
(cmpptr) EXTERNAL TEXT POINTER;  %text ptr. to compiler link%
                                                                    1B2Q
(relptr) EXTERNAL TEXT POINTER;  %text ptr. to rel file link%
                                                                    1B2R
(gi) EXTERNAL; % so we dont get auto-logged out %                   1B2S
(loaded) EXTERNAL = FALSE;     % so we only load once %           1B2T
(relname) EXTERNAL STRING [200]; % rel file name for compiles
%                                                                   1B2U
(qpfilename) EXTERNAL STRING [200];                                 1B2V
(fname) EXTERNAL STRING [70];                                       1B2W
(incfile) EXTERNAL STRING = "(include.sg,)";                        1B2X
   %include sequence generator file%
(supfile) EXTERNAL STRING = "(relnine, libsup.rel,)";               1B2Y
   %library support routines file%
(compiler) EXTERNAL STRING = "<arcsubsys, 1109, >";                 1B2Z
   %name of compiler for LIBRARIAN generated COMPILE
   statements%
% table combinations                NN  NY  NY  MN  MM  MY  YN
YM  YY %
   (ortable) EXTERNAL = ( -1,  0,  1,  0,  0,  1,  1,  1,  1);
                                                                    1B2A@1
   (andtable) EXTERNAL = ( -1, -1, -1, -1,  0,  0, -1,  0,
   1);                                                              1B2A@2
   (xortable) EXTERNAL = ( -1,  0,  1,  0,  0,  0,  1,  0,
   -1);                                                             1B2A@3
% qualifiers & command word strings %
   (csever) EXTERNAL STRING = "everything";                         1B2AA1
   (cslist) EXTERNAL STRING = "list";                               1B2AA2
   (cscomp) EXTERNAL STRING = "compile";                            1B2AA3
   (cspfil) EXTERNAL STRING = "print-file";                         1B2AA4
   (cssysg) EXTERNAL STRING = "sysguide";                           1B2AA5
   (csrunf) EXTERNAL STRING = "runfile";                            1B2AA6
   (csindx) EXTERNAL STRING = "index";                              1B2AA7
   (csupdt) EXTERNAL STRING = "update";                             1B2AA8
   (cscond) EXTERNAL STRING = "conditional";                        1B2AA9
   (csincl) EXTERNAL STRING = "include";                           1B2AA10
   (csptyp) EXTERNAL STRING = "print-typescript";                  1B2AA11
   (csplib) EXTERNAL STRING = "print-library-branch";              1B2AA12
   (cspind) EXTERNAL STRING = "print-index";                       1B2AA13
   (cspsys) EXTERNAL STRING = "print-sysguide";                    1B2AA14
   %(csdtch) EXTERNAL STRING = "detach"; %%not in NLS10%
   (cscomm) EXTERNAL STRING = "comment";                           1B2AA16
   (csstat) EXTERNAL STRING = "status";                            1B2AA17
   (csoctl) EXTERNAL STRING = "octal";                             1B2AA18
   (csdcml) EXTERNAL STRING = "decimal";                           1B2AA19
   (csdate) EXTERNAL STRING = "date";                              1B2AA20
   (csoutp) EXTERNAL STRING = "output";                            1B2AA21
   (csrept) EXTERNAL STRING = "repeat";                            1B2AA22
   (csignr) EXTERNAL STRING = "ignore";                            1B2AA23
   (csabrt) EXTERNAL STRING = "abort";                             1B2AA24
   (cslogo) EXTERNAL STRING = "logout";                            1B2AA25
   (cscopyprinter) EXTERNAL STRING = "copyprinter";                1B2AA26
   %(cswait) EXTERNAL STRING = "wait";%
```

```
    (never) EXTERNAL STRING = "never";                          1B2AA28
    (default) EXTERNAL STRING = "default";                      1B2AA29
    (always) EXTERNAL STRING = "always";                        1B2AA30
    (for) EXTERNAL STRING = "for";                              1B2AA31
    (fget) EXTERNAL STRING = "fget";                            1B2AA32
    (fand) EXTERNAL STRING = "fand";                            1B2AA33
    (fxor) EXTERNAL STRING = "fxor";                            1B2AA34
    (fadd) EXTERNAL STRING = "fadd";                            1B2AA35
    (fsub) EXTERNAL STRING = "fsub";                            1B2AA36
    (fsett) EXTERNAL STRING = "fsett";                          1B2AA37
    (fsetf) EXTERNAL STRING = "fsetf";                          1B2AA38
    (fsetm) EXTERNAL STRING = "fsetm";                          1B2AA39
    (fnot) EXTERNAL STRING = "fnot";                            1B2AA40
    (tft) EXTERNAL STRING = "tft";                              1B2AA41
    (tff) EXTERNAL STRING = "tff";                              1B2AA42
    (tfm) EXTERNAL STRING = "tfm";                              1B2AA43
    (tfnt) EXTERNAL STRING = "tfnt";                            1B2AA44
    (tfnf) EXTERNAL STRING = "tfnf";                            1B2AA45
    (tfnm) EXTERNAL STRING = "tfnm";                            1B2AA46
    (tfe) EXTERNAL STRING = "tfe";                              1B2AA47
    (tfne) EXTERNAL STRING = "tfne";                            1B2AA48
    (tfg) EXTERNAL STRING = "tfg";                              1B2AA49
    (tfng) EXTERNAL STRING = "tfng";                            1B2AA50
    (tfl) EXTERNAL STRING = "tfl";                              1B2AA51
    (tfnl) EXTERNAL STRING = "tfnl";                            1B2AA52
    (writedate) EXTERNAL STRING = "writedate";                  1B2AA53
    (readdate) EXTERNAL STRING = "readdate";                    1B2AA54
    (createdate) EXTERNAL STRING = "createdate";                1B2AA55
  % main branch identifiers %
    (libfile) EXTERNAL STRING = "LF:";                           1B2AB1
    (srcfil) EXTERNAL STRING = "SF:";                            1B2AB2
    (runfil) EXTERNAL STRING = "RF:";                            1B2AB3
    (newproc) EXTERNAL STRING = "NP:";                           1B2AB4
    (cmdbra) EXTERNAL STRING = "CB:";                            1B2AB5
    (libhis) EXTERNAL STRING = "Library History & Summaries";
                                                                 1B2AB6
  % builtin and user flag strings %
    (dindex) EXTERNAL = 0;                                       1B2AC1
    (nusrflgs) EXTERNAL CONSTANT = 40;                           1B2AC2
    (uflag) EXTERNAL STRING = "uf";                              1B2AC3
    (bvar1) EXTERNAL STRING = "filncmp";                         1B2AC4
    (bvar2) EXTERNAL STRING = "filcmp";                          1B2AC5
    (bvar3) EXTERNAL STRING = "oldfile";                         1B2AC6
    (bvar4) EXTERNAL STRING = "newfile";                         1B2AC7
    (bvar5) EXTERNAL STRING = "cmpok";                           1B2AC8
    (bvar6) EXTERNAL STRING = "cmperr";                          1B2AC9
    (bvar7) EXTERNAL STRING = "noerrors";                       1B2AC10
    (bvar8) EXTERNAL STRING = "totalerrors";                    1B2AC11
    (bvar9) EXTERNAL STRING = "nopc";                           1B2AC12
    (bvar10) EXTERNAL STRING = "filepc";                        1B2AC13
    (bvar11) EXTERNAL STRING = "infrun";                        1B2AC14
    (bvar12) EXTERNAL STRING = "infnrun";                       1B2AC15
    (bvar13) EXTERNAL STRING = "infok";                         1B2AC16
    (bvar14) EXTERNAL STRING = "infnok";                        1B2AC17
    (bvar15) EXTERNAL STRING = "numcompiles";                   1B2AC18
    (bvar16) EXTERNAL STRING = "numprints";                     1B2AC19
```

```
    (bvar17) EXTERNAL STRING = "numupdates";                1B2AC20
    (bvar18) EXTERNAL STRING = "numindexes";                1B2AC21
    (bvar19) EXTERNAL STRING = "numsysguides";              1B2AC22
    (bvar20) EXTERNAL STRING = "numconditionals";           1B2AC23
    (bvar21) EXTERNAL STRING = "numincludes";               1B2AC24
    (bvar22) EXTERNAL STRING = "numinferiors";              1B2AC25
    (bvar23) EXTERNAL STRING = "numrcompiles";              1B2AC26
    (bvar24) EXTERNAL STRING = "numicompiles";              1B2AC27
    (bvar25) EXTERNAL STRING = "numokinferiors";            1B2AC28
    (bvar26) EXTERNAL STRING = "numokcompiles";             1B2AC29
    (bvar27) EXTERNAL STRING = "sourcedate";                1B2AC30
    (bvar28) EXTERNAL STRING = "reldate";                   1B2AC31
    (bvar29) EXTERNAL STRING = "lpdate";                    1B2AC32
    (bvar30) EXTERNAL STRING = "lxdate";                    1B2AC33
    (bvar31) EXTERNAL STRING = "ludate";                    1B2AC34
    (bvar32) EXTERNAL STRING = "lidate";                    1B2AC35
    (bvar33) EXTERNAL STRING = "lndate";                    1B2AC36
    (bvar34) EXTERNAL STRING = "lcdate";                    1B2AC37
    (bvar35) EXTERNAL STRING = "libdate";                   1B2AC38
    (bvar36) EXTERNAL STRING = "inxdate";                   1B2AC39
    (bvar37) EXTERNAL STRING = "numpindices";               1B2AC40
    (bvar38) EXTERNAL STRING = "numpsysguide";              1B2AC41
    (bvar39) EXTERNAL STRING = "numlocked";                 1B2AC42
    (bvar40) EXTERNAL STRING = "anyaction";                 1B2AC43
    SET EXTERNAL nbnflgs= 40;   % number of builtin flags %
    (bvarstr) EXTERNAL = (                                  1B2AC45
       $bvar1, $filncmp,
       $bvar2, $filcmp,
       $bvar3, $oldfile,
       $bvar4, $newfile,
       $bvar5, $cmpok,
       $bvar6, $cmperr,
       $bvar7, $noerrors,
       $bvar8, $totalerrors,
       $bvar9, $nopc,
       $bvar10, $filepc,
       $bvar11, $infrun,
       $bvar12, $infnrun,
       $bvar13, $infok,
       $bvar14, $infnok,
       $bvar15, $numcompiles,
       $bvar16, $numprints,
       $bvar17, $numupdates,
       $bvar18, $numindexes,
       $bvar19, $numsysguides,
       $bvar20, $numconditionals,
       $bvar21, $numincludes,
       $bvar22, $numinferiors,
       $bvar23, $numrcompiles,
       $bvar24, $numicompiles,
       $bvar25, $numokinferiors,
       $bvar26, $numokcompiles,
       $bvar27, $sourcedate,
       $bvar28, $relfiledate,
       $bvar29, $lpdate,
       $bvar30, $lxdate,
```

```
        $bvar31, $ludate,
        $bvar32, $lidate,
        $bvar33, $lndate,
        $bvar34, $lcdate,
        $bvar35, $libdate,
        $bvar36, $inxdate,
        $bvar37, $numpindices,
        $bvar38, $numpsysguide,
        $bvar39, $numlocked,
        $bvar40, $anyaction );
    % flags %
      % the following are the command word driving variables and
      they are interpretted as follows:
        < 0: don't perform the specified action under any
        condition
        = 0: perform action if source file write date indicates
        its new
        > 0: perform action regardless of write date of source
        file %
      % builtin flags %
        (anyaction) EXTERNAL;            % 1 If any actions or
        errors %                                          1B2AD2A
        (numlocked) EXTERNAL;            % # files locked by
        someone else %                                    1B2AD2B
        (numpindices) EXTERNAL;  % # indices printed %     1B2AD2C
        (numpsysguide) EXTERNAL; % 1 if sysguide printed; 0
        otherwise %                                       1B2AD2D
        (lpdate) EXTERNAL;               % date librarian last
        printed file %                                    1B2AD2E
        (lxdate) EXTERNAL;               % date librarian last
        indexed file %                                    1B2AD2F
        (ludate) EXTERNAL;               % date librarian last
        updated file %                                    1B2AD2G
        (lidate) EXTERNAL;               % date librarian last
        counted includes %                                1B2AD2H
        (lndate) EXTERNAL;               % date librarian last
        counted conditionals %                            1B2AD2I
        (lcdate) EXTERNAL;               % date librarian last
        compiled file %                                   1B2AD2J
        (libdate) EXTERNAL;              % date librarian last
        run %                                             1B2AD2K
        (sourcedate) EXTERNAL;           % source file date %
                                                          1B2AD2L
        (relfiledate) EXTERNAL;  % rel file date %        1B2AD2M
        (inxdate) EXTERNAL;              % index file date %
                                                          1B2AD2N
        (numokcompiles) EXTERNAL;        % # successful compiles
        %                                                 1B2AD2O
        (numrcompiles) EXTERNAL; % # regular compiles done %
                                                          1B2AD2P
        (numicompiles) EXTERNAL; % # include compiles done %
                                                          1B2AD2Q
        (numcompiles) EXTERNAL;  % # compiles done %      1B2AD2R
        (numprints) EXTERNAL;            % # files printed %
                                                          1B2AD2S
        (numupdates) EXTERNAL;           % # files updated %
```

```
                                                                    1B2AD2T
    (numindexes) EXTERNAL;              % # files indexed %
                                                                    1B2AD2U
    (numsysguides) EXTERNAL; % # indices copied to sysguide
    %                                                               1B2AD2V
    (numconditionals) EXTERNAL;         % # files for which
    conditionals counted %                                          1B2AD2W
    (numincludes) EXTERNAL;  % # files for which includes
    counted %                                                       1B2AD2X
    (numinferiors) EXTERNAL; % # inferior forks run %  1B2AD2Y
    (numokinferiors) EXTERNAL;          % # inferior forks run
    successfully %                                                  1B2AD2Z
    (infnok) EXTERNAL;                  % NOT infok %    1B2AD2A@
    (infok) EXTERNAL;                   % inferior ran ok %
                                                                    1B2AD2AA
    (infnrun) EXTERNAL;                 % NOT infrun %   1B2AD2AB
    (infrun) EXTERNAL;                  % inferior has been run
    %                                                               1B2AD2AC
    (filepc) EXTERNAL;                  % TRUE > file has a pc
    which is ours %                                                 1B2AD2AD
    (nopc) EXTERNAL;            % NOT filepc %                    1B2AD2AE
    (totalerrors) EXTERNAL;  % total number of compile
    errors detected %                                               1B2AD2AF
    (noerrors) EXTERNAL;                % TRUE > no errors in
    any compile %                                                   1B2AD2AG
    (cmperr) EXTERNAL;                  % # errors during
    compile %                                                       1B2AD2AH
    (cmpok) EXTERNAL;                   % TRUE > no errors
    during compile %                                                1B2AD2AI
    (newfile) EXTERNAL;                 % TRUE > source more
    recent than rel %                                               1B2AD2AJ
    (oldfile) EXTERNAL;                 % NOT newfile %  1B2AD2AK
    (filcmp) EXTERNAL;                  % TRUE > file was
    compiled %                                                      1B2AD2AL
    (filncmp) EXTERNAL;                 % NOT filcmp %   1B2AD2AM
(userflags) EXTERNAL [nusrflgs];        % user settable flags %
                                                                      1B2AD3
(dates) EXTERNAL [2];                   % used for referencing
file dates %                                                        1B2AD4
(gabort) EXTERNAL;  % abort processing %                          1B2AD5
(grepeat) EXTERNAL; % repeat branch %                             1B2AD6
(sdrive) EXTERNAL;                                                1B2AD7
(grunfile) EXTERNAL;            % run an inferior %               1B2AD8
(gignore) EXTERNAL; % ignore this branch %                        1B2AD9
%(gdetach) EXTERNAL;           %% run detached if set -- not
in NLS10 %
(goutput) EXTERNAL; % > 0 implies jfn for redirected output
file %                                                              1B2AD11
(gsysgd) EXTERNAL;  % governs sysgds for individual files %
                                                                    1B2AD12
(gconcnt) EXTERNAL; % governs conditional counts for
individual files %                                                  1B2AD13
(ginccnt) EXTERNAL; % governs include counts for individual
files %                                                             1B2AD14
(gprint) EXTERNAL;  % governs printing of individual files
%                                                                   1B2AD15
```

```
    (gptypescript) EXTERNAL;     % governs printing of
    typescript file %                                          1B2AD16
    (gplibbranch) EXTERNAL;      % governs printing of library
    branch %                                                   1B2AD17
    (gcopyprinter) EXTERNAL;  %governs copying of output to
    printer directory%                                         1B2AD18
    (gpindex) EXTERNAL; % governs printing of index files %
                                                               1B2AD19
    (gpsysguide) EXTERNAL;       % governs printing of sysguide
    file %                                                     1B2AD20
    (guptype) EXTERNAL; % governs updating type of individual
    files %                                                    1B2AD21
       % = 1: update old; = 2: update new; = 3: update compact
       %
    (gupdate) EXTERNAL; % governs updating of individual files
    %                                                          1B2AD22
    (gcompile) EXTERNAL;         % governs compiling of
    individual files %                                         1B2AD23
    (gsupersysgd) EXTERNAL;      % governs full sysgd generation
    %                                                          1B2AD24
    (glogout) EXTERNAL; % logout when done %                   1B2AD25
    %(gwait) EXTERNAL;           %% wait until time to run %
    (edrive) EXTERNAL;                                         1B2AD27
% constants for making x-routine arguments %
    (typwrtr) EXTERNAL CONSTANT = 52;  %should use declaration
    in (nine,psedt2)%                                          1B2AE1
    (dsply)EXTERNAL CONSTANT = 51;  %should use declaration in
    (nine,psedt2)%                                             1B2AE2
    (argxcw) EXTERNAL CONSTANT = 1;  %command word argument
    type %                                                     1B2AE3
    (argxvs) EXTERNAL CONSTANT = 2;  %viewspec argument type %
                                                               1B2AE4
%%                                                             1B2AF
```

```
% Procedures %
  (gtdate) PROCEDURE % get processing date from passed stid %
                                                                  1B3A
    % FORMALS %
      ( flstid);        % stid of concerned statement %    1B3A1A
    % LOCALS %
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING locstr[100];
    tp1 _ flstid;   tp1[1] _ 1;
    IF NOT FIND tp1 > [""(("] [":] ^tp1 < $PT > UL $3ULD CH $NP
    ^tp1 ["))"] < 2CH $NP ^tp2 THEN RETURN( 0);
    *locstr* _ tp1 tp2, 0;
    IF locstr.L <= 1 THEN RETURN( 0);
    IF NOT SKIP !idtim( 18M6 .V ($locstr + 1), 0) THEN R2 _ 0;
    RETURN( R2);
    END.
    %%                                                            1B3A10
```

```
(loadall) PROCEDURE % library init: load include & lib support
file %                                                          1B3B
   % FORMALS %
     ;
   % LOCALS %
     LOCAL ainclude;
     LOCAL TEXT POINTER tp1;
     LOCAL parray[40];
     (supload) _ FALSE;  %TRUE when loading support%      1B3B2D
     (retvalue) _ TRUE;                                   1B3B2E
        %return value -- set to FALSE if can't load include &
        lib support file%
     (emsg) STRING [2000];  %error message%               1B3B2F
   INVOKE (catloadall, retloadall);
   IF NOT FIND SF( *upgnms*) ["INCLUDE"] THEN
     BEGIN
     dismes( 1, $"Loading INCLUDE Sequence Generator");
     tp1 _ $incfile;   tp1.stastr _ TRUE;
        tp1[1] _ 1;
     lnkprs( $tp1, $parray);
     ldprog( $parray, FALSE %no loader dismes%, $emsg);
     END;
   IF NOT FIND SF( *upgnms*) ["LIBSUP"] THEN
     BEGIN
     supload _ TRUE;
     dismes( 1, $"Loading LIBRARY Support Routines");
     tp1 _ $supfile;   tp1.stastr _ TRUE;
        tp1[1] _ 1;
     lnkprs( $tp1, $parray);
     ldprog( $parray, FALSE %no loader dismes%, $emsg);
     END;
   emsg.L _ 0;  %don't display undefines%
   (retloadall):                                         1B3B7
   DROP (catloadall);
   IF emsg.L THEN dismes(1, $emsg);  % display error message %
   RETURN( retvalue);
   (catloadall) CATCHPHRASE();                           1B3B11
     BEGIN
     CASE SIGNALTYPE OF
        = aborttype :
          BEGIN  %failure in loading%
          DISABLE (catloadall);
          IF supload THEN
             *emsg* _ "Unable to load LIBRARY Support
             Routines"
          ELSE *emsg* _ "Unable to load INCLUDE Sequence
          Generator";
          retvalue _ FALSE;  % return FALSE %
          TERMINATE;
          END;
        ENDCASE;
     CONTINUE;
     END;
   END.
   %%                                                    1B3B13
```

```
(loadchk) PROCEDURE % check to see if INCLUDE and SUPPORT
loaded %                                                    1B3C
   % FORMALS %
     ;
   % LOCALS %
   IF NOT FIND SF( *upgnms*) ["INCLUDE"] THEN
      BEGIN
      dismes( 1, $"User Program Buffer FOULUP; Please DELETE
      ALL programs & reload LIBRARY SUBSYS");
      RETURN( FALSE);
      END;
   IF NOT FIND SF( *upgnms*) ["LIBSUP"] THEN
      BEGIN
      dismes( 1, $"User Program Buffer FOULUP; Please DELETE
      ALL programs & reload LIBRARY SUBSYS");
      RETURN( FALSE);
      END;
   RETURN( TRUE);
   END.
   %%                                                       1B3C7
```

```
(xlbevtop) PROCEDURE % xroutine for process command %      1B3D
   % FORMALS %
      (destlist REF);  %branch entity%
   % LOCALS %
      LOCAL oldname;
      LOCAL oldmode;
      (destination) REF;                                   1B3D2C
      (emsg) REF;  %for abort error message%               1B3D2D
      (arglist) LIST [2];  %argument for xsimulate%        1B3D2F
   % invoke catchphrase %
      INVOKE (catxlbevtop, retxlbevtop);
         % catch all ABORTS so that middle-end does not do an
         error return.  An error return would terminate the
         command and possibly leave the terminal in the wrong
         mode %
   % prepare parameters %
      &destination _ #destlist#[tppair];  %adr. of text ptr
      block%
   % do the command %
      IF NOT loadchk() THEN RETURN( FALSE);
      !gtad();
         libsrttime _ R1;      % library processing start time
         %
      !getnm();
         oldname _ R1;
      R1 _ 606354B6;  % sixbit for PSL %
         !setnm();
      gi _ 0;
      topalmost( destination);
      IF gi THEN
         BEGIN
         !dic( 4B5, 000010B6);
         %!iit( 4B5, 000010B6, 0);%
            R1 _ 4B5;   R2 _ 000010B6;   R3 _ 0; !JSYS 630B;
         chntab[14] _ gi;
         END;
      !setnm( oldname);
   (retxlbevtop):                                          1B3D6
   RETURN;
      (catxlbevtop) CATCHPHRASE(: &emsg);                  1B3D7A
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL
            = aborttype :
               BEGIN  %prevent middle-end from catching ABORT%

               DISABLE (catxlbevtop);
               IF &emsg AND emsg.L THEN dismes(1, &emsg)
               ELSE  %no error message -- tell user something%
                  dismes(1, $"Error in Process command -- no
                  explanation available");
               TERMINATE;  %goes to retxlbevtop%
               END;
            ENDCASE;
         CONTINUE;
```

```
        END;
    END.
%%                                                    18309
```

```
(xlchkadr)   % CL: ;  dummy xroutine for checking address %
PROCEDURE (source REF);                                          1B3E
   % Procedure description
      FUNCTION
         This procedure does not do anything.  It is called
         from the Process command so that argument conversion
         will check the address before the terminal mode gets
         changed.  If the address is not valid, this procedure
         will never be reached and the failure in the argument
         conversion procedure will cause the command to be
         aborted
      ARGUMENTS
         source:  address for library branch
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
   % Don't do anything %
   % Return %
      RETURN;
   END.

(xldelhis) PROCEDURE % xroutine for deleting history %      1B3F
   % FORMALS %
      (timelist REF LIST,      % text entity with time to
      delete before%
      destlist REF LIST);      % branch entity %
   % LOCALS %
      (destination) REF;  %adr. of text pointer block%    1B3F2A
      (timeptr) REF;  %adr. of text pointer block%        1B3F2B
      LOCAL histime;
      LOCAL hisstid;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING locstr [40];
   % prepare parameters %
      &destination  _ #destlist#[tppair];
      &timeptr _ #timelist#[tppair];
   % do command %
      IF NOT loadchk() THEN RETURN( FALSE);
      % get internal date and time %
         tp1 _ timeptr;                tp1[1] _ [&timeptr +
         1];
         tp2 _ [&timeptr + 2]; tp2[1] _ [&timeptr + 3];
         *locstr* _ tp1 tp2, 0;   BUMP DOWN locstr.L;
         IF NOT SKIP !idtim( 18M6 .V ($locstr + 1), 0) THEN
            BEGIN
            *locstr* _ *locstr*, " 12:00:00AM", 0;
            IF NOT SKIP !idtim( 18M6 .V ($locstr + 1), 0) THEN
               err( $"Illegal time specified");
            END;
         histime _ R2;
      % get stid of start %
```

```
            topstid _ destination;
         % take care of screen %
            clist( 15 %ctlcfm%, topstid.stfile, 0);
            dpset( dspstrc, topstid, endfil, dpstp(topstid));
         % loop through form deleting appropriately %
            cfstid _ getsub( topstid);
            WHILE cfstid # topstid DO
               BEGIN
               IF (srcstid _ getsub( cfstid)) # cfstid THEN
                  BEGIN
                  WHILE srcstid # cfstid DO
                     BEGIN
                     hisstid _ getsub( srcstid);
                     WHILE (hisstid # srcstid) AND (gtdate(
                     hisstid) >= histime) DO
                        hisstid _ getsuc( hisstid);
                     IF hisstid # srcstid THEN
                        cdelgro( hisstid, getail( hisstid), 0,
                        0);
                     srcstid _ getsuc( srcstid);
                     END;
                  END;
               cfstid _ getsuc( cfstid);
               END;
         % take care of screen %
            clupdt();
      RETURN;
      END.
      %%                                                    1B3F7
```

```
(xlicompile) PROCEDURE % xroutine compile file with includes %
                                                                1B3G
   %FORMALS%
      (entity REF,      %entity command word%
      location REF,     %selected location to begin%
      compname REF,     %name of compiler%
      flname REF,                          %name of output file%
      dname REF);       %diagnostic destination%
   LOCAL
      da,
      ret,
      savusq;  %save location of user sequence generator %
   LOCAL STRING
      locstr[100];
      (vs) [4];  %viewspec block%                               1B3G39
   REF da, tptr2;
   % do command %
      IF NOT loadchk() THEN RETURN( FALSE);
      &da _ lda();
      inccount _ ret _ 0;
      savusq _ da.dausqcod := $include;
      INVOKE(catxlicompile, xicfix);
      makexarg(argxvs, $vs, $"0", da.dawid);
      ret _ xcompile( &entity, &location, &compname, &flname,
      &dname, $vs, da.dawid);
      (xicfix):                                                 1B3G5H
      DROP (catxlicompile);
      da.dausqcod _ savusq;
      CASE inccount OF
         = 0: *locstr* _ "No INCLUDEs encountered", CR, LF;
         ENDCASE
            *locstr* _ STRING( inccount), " INCLUDEs processed
            successfully", CR, LF;
      dismes( 1, $locstr);
   RETURN( ret );
   % catchphrases %
      (catxlicompile) CATCHPHRASE();                            1B3G7A
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               TERMINATE;
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                           1B3G9
```

```
(xiiinit) PROCEDURE; % xroutine library initialization: load
include %                                                    1B3H
   % FORMALS %
   % LOCALS %
   % Load support and include sequence generator if necessary
   %
      IF NOT loaded THEN
         BEGIN
         IF NOT loadall() THEN RETURN( FALSE);
         loaded _ TRUE;
         END
      ELSE IF NOT loadchk() THEN RETURN( FALSE);
   RETURN;
   END.
   %%                                                        1B3H6
```

```
(xlreset)   % CL: ;  xroutine to reset the display %
PROCEDURE (dpytype REF, sourceptr REF);                          1B3I
   % Procedure description
     FUNCTION
        Call "xsimulate" to go back to display mode and
        "dpset" to refresh the screen.  This procedure is
        called from the Process command.
     ARGUMENTS
        dpytype:  DISPLAY command word
        sourceptr:  source pointer for Process command
     RESULTS
        proc-value
     NON-STANDARD CONTROL
        none
     GLOBALS
        none
     %
   % Declarations %
     (source) REF;                                               1B3I2A
   % prepare argument %
     &source _ ELEM #sourceptr#[tppair];
   % Set up for screen refresh %
     dpset(dspallf, source, endfil, erdfil);
   % Go back to display mode %
     xsimulate (&dpytype);
   % Return %
     RETURN;
   END.

   %%                                                            1B3I8
```

FINISH

```
%libsup% FILE lsupport % (arcsubsys, 1109,)  (relnine, libsup.rel,)
%                                                                    1C
   NOMESS
   ALLOW!
   % DECLARATIONS %
      % strings for 'finder' -- can't be local because stack is
      small with sequence generator %
         (fdrstmnt) EXTERNAL STRING [2000];                          1C3A1
         (fdrstnum) EXTERNAL STRING [20];                            1C3A2
         (fdrcomment) EXTERNAL STRING [1500];                        1C3A3
         (fdrparams) EXTERNAL STRING [500];                          1C3A4
         (fdrtype) EXTERNAL STRING [50];                             1C3A5
         (fdrwidth) EXTERNAL STRING  [50];                           1C3A6
   (topalmost) PROCEDURE % process main branch %                      1C4
      % FORMALS %
         (libstid);                                                  1C4A1
      % LOCALS %
         LOCAL svint;
         LOCAL sgoutput;
         LOCAL i;
         LOCAL pcm;
         LOCAL init;
         LOCAL dirnum;
         LOCAL fl;   REF fl;
         LOCAL cp;
         LOCAL oldcp;
         LOCAL diff;
         LOCAL temp;
         LOCAL sav3;
         (emsg) REF;  %error message at ABORT%                      1C4B13
         LOCAL TEXT POINTER tp1, tp2;
         LOCAL darray[100];
         LOCAL STRING locstr[150];
         LOCAL STRING l2str[10];
         LOCAL STRING l3str[300];
         LOCAL STRING lkstr[100];
         LOCAL STRING messtr[1000];
      topstid _ libstid;
      svint _ FALSE;
      % initialize driving variables to default values %
         FOR i _ 0 UP UNTIL = nusrflgs DO userflags[i] _ -1;
         numlocked _ 0;                % 0 files locked by someone
         else %
         numpindices _ 0;     % 0 indices printed %
         numpsysguide _ 0;    % 0 sysguides printed %
         numcompiles _ 0;     % 0 compiles done %
         numokcompiles _ 0;   % 0 successful compiles done %
         numrcompiles _ 0;    % 0 regular compiles done %
         numicompiles _ 0;    % 0 include compiles done %
         numprints _ 0;                % 0 files printed %
         numupdates _ 0;      % 0 files updated %
         numindexes _ 0;      % 0 files indexed %
         numsysguides _ 0;    % 0 indices copied to sysguide %
         numconditionals _ 0;          % 0 files for which
         conditionals counted %
         numincludes _ 0;     % 0 files for which includes counted %
```

```
        numinferiors _ 0;    % 0 inferior forks run %
        numokinferiors _ 0; % 0 inferior forks run successfully %
        infrun _ -1;         % no inferiors run yet %
        infnrun _ - infrun;
        infok _ 1;  % inferior ran ok %
        infnok _ - infok;
        grunfile _ 1;
        totalerrors _ 0;
        anyaction _ 0;
        noerrors _ 1;
        gignore _ gabort _ grepeat _ -1;
        gsysgd _ 0; % do sysgd for each file %
        gconcnt _ 0;         % count conditionals for each file %
        ginccnt _ 0;         % count includes for each file %
        gprint _ 0; % print each file as appropriate %
        gpindex _ 0;         % print index files as appropriate %
        gptypescript _ 0;    % print typescript file as appropriate
        %
        gplibbranch _ 0;     % print library branch as appropriate %
        gcopyprinter _ 0;    % copy output to printer as appropriate
        %
        gpsysguide _ 0;      % print sysguide as appropriate %
        gsupersysgd _ 0;     % generate full sysgd %
        gcompile _ 0;        % gcompile each file %
        gupdate _ 0;         % default update each file %
        guptype _ 2;         % update new each file %
        %gdetach _ -1;        %% run attached --  detached not in
        NLS10 %
        glogout _ -1;        % dont logout when done %
        gwait _ 0;  % run immediately %
        !gpjfn( 400000B);    % get starting primary jfns %
           goutput _ srtjfns _ R2;
      % parse top statement - get stid for sysguide %
        tp2 _ topstid;   tp2[1] _ 1;
        IF NOT FIND tp2 > [*libfile*] ^tp2 THEN
           BEGIN
           dismes( 1, $"Invalid Plex header detected, Library
           processing aborted");
           RETURN( FALSE);
           END;
        IF NOT sprstid _ gmstid( topstid, cp _ tp2[1], TRUE, TRUE)
        THEN
           BEGIN
           dismes( 1, $"Invalid Plex header detected, Library
           processing aborted");
           RETURN( FALSE);
           END;
      % parse top statement - abort sysguide locked by some else %
        &fl _ flntadr( sprstid.stfile);
        IF fl.fllock AND (NOT fl.flpart) THEN
           BEGIN
           !gtfdb( fl.florig, 186 + 24B, $R3);
           init _ R3.lkinit;
           dirnum _ R3.lkdirn;
           lockid( $lkstr, dirnum, init);
           IF FIND SE(*lkstr*) ^tp2 [".()] SP 1$ULD ^tp1 THEN
```

```
            *lkstr* _ + tp1 tp2;
         *messtr* _
            ": LOCKED BY ", *lkstr*, "; Library processing
            aborted";
         wrtmsg( 0, $messtr);
         gsupersysgd _ -1;
         clsfile( sprstid := 0);
         RETURN( FALSE);
         END;
      % get some stids; abort if problems %
       *locstr* _ NULL;
       CASE crehis() OF
         < 0:
            *locstr* _ "Library Branch Substructure missing:
            Library processing aborted";
         = 0: NULL;
         > 0:
            *locstr* _ "Unable to create Library History Branch:
            Library processing aborted";
         ENDCASE;
       IF locstr.L THEN
         BEGIN
         dismes( 1,$locstr);
         IF sprstid THEN clsfile( sprstid := 0);
         RETURN( FALSE);
         END;
      % initialize printing file %
       IF NOT initprnt() THEN
         BEGIN
         dismes( 1, $"Unable to initialize printing file; Library
         processing aborted");
         IF sprstid THEN clsfile( sprstid := 0);
         RETURN( FALSE );
         END;
      % parse top statement - tell user we are starting %
       *locstr* _ "(( ))";
       tp1 _ $locstr;   tp1.stastr _ TRUE;   tp1[1] _ 1;
       rplpdate( tp1, $"Library processing started by");
       dismes( 1, $locstr);
       tp2 _ tp1;   tp2[1] _ locstr.L + 1;
       lhsstid _ cinssta( lhsstid, -1, $tp1, $tp2);
       ccopsta( lhsstid, -1, topstid, 0, 0);
      % parse top statement - initialize builtin flags %
       % get date librarian was last run %
         IF getftl( lhsstid) THEN libdate _ 0
         ELSE libdate _ gtdate( getsuc( lhsstid));
       % indicate whether or not we have a pc %
         filepc _ IF fl.flpart THEN 1 ELSE -1;
         nopc _ - filepc;
       % indicate any compile errors %
         cmperr _ -1;
         cmpok _ 1;
       % indicate if source newer than rel (ie has a pc in this
       case) %
         newfile _ filepc;
         oldfile _ - newfile;
```

```
        % indicate not compiled yet %
          filcmp _ -1;
          filncmp _ 1;
      % parse top statement - parse pre commands %
        IF NOT preparsecmd( topstid, oldcp _ cp : cp) THEN
          BEGIN
          dismes( 1, $"Unable to parse pre commands; Processing
          aborted");
          IF sprstid THEN clsfile( sprstid := 0);
          RETURN( FALSE);
          END;
        IF NOT sprstid THEN gsupersysgd _ -1;
        IF goutput # srtjfns THEN
          BEGIN
          jfntostr( goutput.RH, $13str, 011110B6 .V 1);
          *13str* _ "Changing primary output file to: ", *13str*;
          dismes( 1, $13str);
          % IF gdetach >= 0 THEN dismes( 1, $"Detaching job"); %
          !spjfn( 4000008, goutput);
          dismes( 1, $locstr);
          END;
        % IF gdetach >= 0 THEN
          BEGIN
          IF goutput = srtjfns THEN
            BEGIN
            dismes( 1, $"Illegal to detach without specifying an
            output file; detach command ignored");
            END
          ELSE
            BEGIN
            dismes( 1, $"Detaching job");
            !dtach();
            END;
          END; %
        IF oldcp # cp THEN
          BEGIN
          diff _ oldcp - cp;
          FOR temp _ $garray + 2 UP 2 UNTIL > $garray + 28 DO
            [temp] _ [temp] - diff;
          END;
          % wait not implemented in NLS10
        IF gwait THEN
          BEGIN
          sav3 _ chntab[3] := 1B6 .V $tclo;
          WHILE gwait > (5 * 60) DO
            BEGIN
            !disms( (5 * 60) * 1000);
            gwait _ gwait - (5 * 60);
            END;
          !disms( (gwait := 0) * 1000);
          !JFCL;
          (t2clo):                                              1C4L6F
          chntab[3] _ sav3;
          IF svint THEN
            BEGIN
            dismes( 1, $"Library processing aborted by user");
```

```
            glogout _ -1;
            GOTO done;
            END
         ELSE
            BEGIN
            rplpdate( tp1, $"Library processing resumed by");
            dismes( 1, $locstr);
            END;
         END;
      %
   % indicate no new file sysgds yet %
      INVOKE(cattopalmost1, top1);
      IF sprstid THEN isysgd( sprstid );
      newsysgdcnt _ 0;
   (top1):  % process each file %                                    1C4N
      DROP (cattopalmost1);
      sysorg _ srcstid _ 0;
      INVOKE (cattopalmost2, next);
      DISABLE (cattopalmost2);  %enable below%
      LOOP
         BEGIN
         % make listing nice %
            *12str* _ CR, LF, CR, LF, CR, LF;
            dismes( 1, $12str);
         % save driving variables state %
            pcm _ FALSE;
            FOR i _ $sdrive UP UNTIL >= $edrive DO
            darray[i-$sdrive] _ [i];
         ENABLE (cattopalmost2);
         % process the branch %
            tp2 _ cfstid;   tp2[1] _ 1;
            IF FIND tp2 > [*srcfil*] ^tp2 THEN dofile( tp2[1])
            ELSE IF FIND tp2 > [*runfil*] ^tp2 THEN runfork(
            tp2[1], FALSE)
            ELSE IF FIND tp2 > [*newproc*] ^tp2 THEN runfork(
            tp2[1], TRUE)
            ELSE IF FIND tp2 > [*cmdbra*] THEN pcmdbra( pcm _
            TRUE);
            DISABLE (cattopalmost2);
         (next):                                                  1C4N5F
         % close some files %
            IF sysorg THEN clsfile ( sysorg := 0);
            IF srcstid THEN clsfile( srcstid := 0);
         % restore driving variables %
            IF pcm := FALSE THEN
               BEGIN
               sgoutput _ darray[ $goutput - $sdrive];
               IF sgoutput # goutput THEN clspjfn( sgoutput.RH);
               END
            ELSE
               BEGIN
               sgoutput _ goutput;
               FOR i _ $sdrive UP UNTIL >= $edrive DO [i] _
               darray[i-$sdrive];
               IF sgoutput # goutput THEN
                  BEGIN
```

```
                !spjfn( 400000B, goutput);
                clspjfn( sgoutput.RH );
                END;
             END;
          % exit if ABORT asked for %
             IF gabort >= 0 THEN EXIT LOOP;
          % move on to next one if there is one (and no ^O) %
             IF (grepeat := -1) < 0 THEN
                BEGIN
                IF getftl( cfstid) THEN EXIT LOOP;
                cfstid _ getsuc( cfstid);
                END;
             IF svint OR (svint _ inptrf := FALSE) THEN EXIT LOOP;
          END;
       (done):  % make listing look nice %                          1C40
          DROP (cattopalmost2);
          *l2str* _ CR, LF, CR, LF, CR, LF;
          dismes( 1, $l2str);
          IF svint OR (svint _ inptrf := FALSE) THEN
             BEGIN
             IF gabort >= 0 THEN
                dismes( 1, $"Library processing ABORTED by command")
             ELSE dismes( 1, $"User terminated library processing");
             dismes( 1, $l2str);
             END;
       % finish up super sysgd if called or delete modifications %
          INVOKE (cattopalmost3, afsy);
          IF sprstid AND NOT (svint OR (svint _ inptrf := FALSE))
          THEN
             BEGIN
             fsuper();
             dismes( 1, $l2str);
             END;
          (afsy):                                                   1C4P3
             DROP (cattopalmost3);
          IF sprstid THEN clsfile( sprstid := 0);
       % set "anyaction" builtin flag %
          anyaction _
             IF numlocked OR numcompiles OR totalerrors OR
             numinferiors OR numupdates OR numprints OR numindexes OR
             numpindices OR numsysguides OR numpsysguide OR
             numincludes OR numconditionals
                THEN 1 ELSE 0;
       % process any post commands %
          IF NOT postparsecmd(topstid, cp) THEN
             BEGIN
             dismes( 1, $"Unable to process post commands");
             dismes( 1, $l2str);
             END;
       % give user final summary %
          IF anyaction THEN
             BEGIN
             *messtr* _ NULL;
             IF numlocked THEN *messtr* _
                "***  ", STRING( numlocked), " FILE(S) LOCKED BY
                SOMEONE ELSE  ***", CR, LF;
```

```
        IF (i _ numcompiles - numokcompiles) OR totalerrors THEN
        *messtr* _
           *messtr*, "***  ", STRING( i), " UNSUCCESSFUL
           COMPILE(S); ", STRING( totalerrors), " TOTAL COMPILE
           ERROR(S)  ***", CR, LF;
        IF (i _ numinferiors - numokinferiors) THEN *messtr* _
        *messtr*,
           "***  ", STRING( i), " UNSUCCESSFUL INFERIOR
           PROCESS(ES)  ***", CR, LF;
        IF messtr.L THEN *messtr* _ *messtr*, CR, LF;
        CASE numupdates OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numupdates), " File(s) updated", CR, LF;
        CASE numcompiles OF
           = 0: NULL;
           = numrcompiles: *messtr* _ *messtr*,
              STRING( numcompiles), " Regular compile(s)", CR,
              LF;
           = numicompiles: *messtr* _ *messtr*,
              STRING( numcompiles), " Include compile(s)", CR,
              LF;
           ENDCASE *messtr* _ *messtr*,
              STRING( numcompiles), " Compile(s) (", STRING(
              numrcompiles), " Regular; ", STRING(
              numicompiles), " Include)", CR, LF;
        CASE numinferiors OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numinferiors), " Inferior process(es)
              run", CR, LF;
        CASE numprints OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numprints), " File(s) printed", CR, LF;
        CASE numindexes OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numindexes), " Index(es) created", CR, LF;
        CASE numpindices OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numpindices), " Index(es) printed", CR,
              LF;
        CASE numsysguides OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              STRING( numsysguides), " Index(es) copied to
              sysguide", CR, LF;
        CASE numpsysguide OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
              "Sysguide printed", CR, LF;
        CASE numincludes OF
           = 0: NULL;
           ENDCASE *messtr* _ *messtr*,
```

```
            STRING( numincludes), " File(s) had its INCLUDEs
            counted", CR, LF;
         CASE numconditionals OF
            = 0: NULL;
            ENDCASE *messtr* _ *messtr*,
               STRING( numconditionals), " File(s) had its
               CONDITIONALs counted", CR, LF;
         END
      ELSE *messtr* _ "No actions performed", CR, LF;
      dismes( 1, $messtr);
      dismes( 1, $12str);
      messtr.L _ messtr.L - 2;
         cnvcrlfteol( $messtr);
         tp1 _ $messtr;   tp1.stastr _ TRUE;   tp1[1] _ 1;
         tp2 _ tp1;   tp2[1] _ messtr.L + 1;
      cinssta( lhsstid, -1, $tp1, $tp2);
   % update library branch file & tell user %
      wrtmsg( -topstid.stfile, $": Updating (new)");
      INVOKE (cattopalmost4, done2);
      cupdfil( topstid.stfile, newversion, 0);
      wrtmsg( -topstid.stfile, $": Done Updating (new)");
      dismes( 1, $12str);
      (done2):                                                  1C4T6
      DROP (cattopalmost4);
   IF NOT (svint OR (svint _ inptrf := 0)) THEN
      BEGIN
      % print the library branch %
         oplib();
      % copy file to arcprinter %
         rnfl();
         dismes( 1, $12str);
      END;
   % tell user we are done %
      tp1 _ $locstr;   tp1.stastr _ TRUE;   tp1[1] _ 1;
      rplpdate( tp1, $"Library processing finished by");
      dismes( 1, $locstr);
   % restore starting primary jfns %
      IF goutput # srtjfns THEN
         BEGIN
         !spjfn( 400000B, srtjfns);
         svint _ inptrf := 0;
         clspjfn( goutput.RH);
         END;
   % logout if asked to %
      IF glogout >= 0 THEN xlogout();
   RETURN( TRUE);
   (tclo):                                                      1C4Z
      svint _ TRUE;
      chntab[3] _ sav3;
      levllc _ $t2clo;
      !debrk();
   % catchphrases %
      (cattopalmost1) CATCHPHRASE();                            1C4A@1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
```

```
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost1);
               dismes( 1, $"Unable to initialize sysgd file,
               Sysguide functions ABORTED");
               gsupersysgd _ gpsysguide _ -1;
               IF sprstid THEN clsfile( sprstid := 0);
               TERMINATE;  %goes to 'top1'%
               END;
            ENDCASE;
         CONTINUE;
         END;
      (cattopalmost2) CATCHPHRASE( : &emsg);                    1C4A@2
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost2);
               IF NOT (svint _ inptrf := FALSE) THEN
                  BEGIN  % not ^O %
                  IF emsg.L THEN wrtmsg( 0, emsg);
                  wrtmsg( 0, $": *** Error while processing");
                  END;
               TERMINATE;  %goes to 'next'%
               END;
            ENDCASE;
         CONTINUE;
         END;
      (cattopalmost3) CATCHPHRASE();                            1C4A@3
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost3);
               wrtmsg( -sprstid.stfile, $": Unable to finish
               processing sysguide");
               dismes( 1, $12str);
               TERMINATE;  %goes to 'afsy'%
               END;
            ENDCASE;
         CONTINUE;
         END;
      (cattopalmost4) CATCHPHRASE( : emsg );                    1C4A@4
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (cattopalmost4);
               IF emsg THEN dismes( 1, emsg);
```

```
                dismes( 1, $12str);
                TERMINATE;  %goes to "done2"%
                END;
            ENDCASE;
        CONTINUE;
        END;
END.
%%                                                          1C4AB
```

```
(clsfile) PROCEDURE % close file we are done with %                    1C5
   % FORMALS %
      ( flstid);  % stid of origin of concerned file %                 1C5A1
   % LOCALS %
   IF NOT flstid.stfile THEN RETURN;
   INVOKE (catclsfile, retclsfile);
   close( flstid.stfile);
   (retclsfile):                                                       1C5F
   DROP (catclsfile);
   RETURN;
   % catchphrases %
      (catclsfile) CATCHPHRASE();                                      1C5I1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
            ENDCASE;
               TERMINATE;
         CONTINUE;
         END;
   END.
   %%                                                                  1C5K
```

```
(clspjfn) PROCEDURE % copy to <*prtdir*> and close a primary
output jfn %                                                    1C6
   % FORMALS %
      (ojfn);                                                   1C6A1
   % LOCALS %
      LOCAL njfn; % jfn of printer copy %
      LOCAL lchar;
      LOCAL tnread;
      LOCAL nread;
      LOCAL bytecnt;
      LOCAL byteptr;
      LOCAL STRING locstr[500];
   % dont copy unless asked for %
      IF gptypescript < 0 THEN
         BEGIN
         IF NOT SKIP !closf( ojfn) THEN NULL;
         RETURN( TRUE);
         END;
   jfntostr( ojfn, $locstr, 001100B6 .V 1);
   *locstr* _ "<, *prtdir*, ">, *initsr*, "-, *locstr*,
   ";P777777", 0;
   IF NOT SKIP !gtjfn( 4B11 .V 1B6, 18M6 .V ($locstr + 1) ) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      RETURN( FALSE );
      END;
   njfn _ R1;
   IF NOT SKIP !openf( njfn, 07B10 .V3B5) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( FALSE );
      END;
   IF NOT SKIP !rfptr( ojfn) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( FALSE );
      END;
   bytecnt _ R2;
   IF NOT SKIP !sfptr( ojfn, 0) THEN
      BEGIN
      IF NOT SKIP !closf( ojfn) THEN NULL;
      IF NOT SKIP !closf( njfn) THEN NULL;
      RETURN( FALSE );
      END;
   tnread _ 0;
   WHILE tnread < bytecnt DO
      BEGIN
      !sin( ojfn, 18M6 .V ($locstr + 1), 499, 37B);
      byteptr _ R2;
      nread _ 499 - R3;
      tnread _ tnread + nread;
      lchar _ .byteptr;
      IF lchar = 37B THEN
         BEGIN
```

```
        lchar _ 15B;
        .byteptr _ lchar;
        lchar _ 12B;^byteptr _ lchar;
        BUMP nread;
        END;
    !sout( njfn, 18M6 .V ($locstr + 1), -nread);
    END;
IF NOT SKIP !delnf( ojfn, 2) THEN NULL;
IF NOT SKIP !closf( ojfn) THEN NULL;
IF NOT SKIP !closf( njfn) THEN NULL;
RETURN( TRUE);
END.
%%                                                    1C6S
```

```
(cmpfile) PROCEDURE % compile file if needed %                    1C7
   % FORMALS %
     ;
   % LOCALS %
     LOCAL savvs;
     LOCAL savvs2;
     LOCAL savca;
     LOCAL savus;
     LOCAL savcsp;
     LOCAL da;  REF da;
     LOCAL errors;
     LOCAL nincludes;
     (vsptr) _ 0;  %set to $vs if INCLUDE's%                      1C7B9
     (retvalue) _ 1 ;                                            1C7B10
     (vs) [4];  %viewspec block%                                 1C7B11
     LOCAL TEXT POINTER tp1, tp2, ta1, ta2;
     LOCAL STRING locstr[200];
   % find out if we need compile %
     CASE gcompile OF
        < 0: RETURN( 0);
        = 0: IF reldate >= srcwrd THEN RETURN( 0);
        ENDCASE;
   % find out if any includes %
     tp1 _ incstid;   tp1[1] _ 1;
     nincludes _ 0;
     IF FIND tp1 > $NP ^tp1 1$D ^tp2 THEN
        BEGIN
        *locstr* _ tp1 tp2;
        nincludes _ VALUE( $locstr );
        END;
   % tell user which file we are starting to compile %
     IF nincludes > 0 THEN *locstr* _ ": Include compiling"
     ELSE *locstr* _ ": Compiling";
     wrtmsg( 0, $locstr);
   % find out and set tp1 to where to start compiling %
     &da _ lda();
     tp1 _ srcstid;   tp1[1] _ 1;
     ta1 _ aarray[1s];   ta1[1] _ aarray[1s+1];
     ta2 _ aarray[1e];   ta2[1] _ aarray[1e+1];
     caddexp( $ta1, $ta2, &da, $tp1)  %sets tp1%;
     DROP (catcmpfile1);
   % do it %
     errors _ 0;
     savcsp _ da.dacsp := tp1;
     savca _ da.dacacode := 0;
     savus _ da.dausqcode := 0;
     savvs2 _ da.davspc2 := 0;
     savvs _ da.davspec := 0;
        da.davspec.vslev _ da.davspec.vstrnc _ da.davspec.vsindf
        _ da.davspec.vsnamf _ -1;
     IF nincludes > 0 THEN
        BEGIN
        da.dausqcod _ $include;
        makexarg(argxvs, $vs, $"0", da.dawid);  %form vs arg%
        vsptr _ $vs;  %use sg vs%
        inccount _ 0;
```

```
        END;
      INVOKE (catcmpfile2, ecmp);
      errors _ ccompile(cwfile, tp1, $cmpptr, $relptr, 0, vsptr,
      &da  );
      (ecmp):  DROP (catcmpfile2);;                                  1C7G10
      da.dacsp _ savcsp;
      da.davspec _ savvs;
      da.davspc2 _ savvs2;
      da.dacacode _ savca;
      da.dausqcod _ savus;
    % tell user of errors or completion %
      IF nincludes > 0 THEN BUMP numicompiles ELSE BUMP
      numrcompiles;
      BUMP numcompiles;
      filcmp _ 1;   filncmp _ -1;
      IF errors <= 0 THEN
         BEGIN
         *locstr* _ ": Done ", *locstr*[3 TO locstr.L];
         BUMP numokcompiles;
         END
      ELSE
         BEGIN
         totalerrors _ errors + totalerrors;
         noerrors _ -1;
         cmperr _ errors;   cmpok _ -1;
         *locstr* _ ": ", STRING( errors), " errors detected
         while ", *locstr*[3 TO locstr.L];
         END;
      wrtmsg( 0, $locstr);
      IF nincludes > 0 THEN
         BEGIN
         *locstr* _ ": ";
         IF inccount > 0 THEN *locstr* _
            *locstr*, STRING( inccount), " INCLUDEs processed
            successfully"
         ELSE *locstr* _ *locstr*, "No INCLUDEs encountered";
         wrtmsg( 0, $locstr);
         END;
    % update relstid with [error] count %
      ccopsta( relstid, -1, relstid, FALSE, FALSE);
      *locstr* _ NULL;
      IF errors > 0 THEN *locstr* _ STRING( errors), " errors; ";
      IF nincludes THEN *locstr* _ *locstr*, "Include ";
      *locstr* _ *locstr*, "Compiled by";
      rplpdate( relstid, $locstr);
      ccopsta( lhsstid, -1, relstid, 0, 0);
    (retcmpfile):                                                    1C7J
    DROP (catcmpfile2);
    RETURN( retvalue );
    % catchphrases %
      (catcmpfile1) CATCHPHRASE();                                   1C7M1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
```

```
            ENDCASE;
               BEGIN
               DISABLE (catcmpfile1);
               wrtmsg( 0, $": Unable to determine compiling start
               address, compile aborted");
               retvalue _ 0;
               TERMINATE;  %goes to "retcmpfile"%
               END;
         CONTINUE;
         END;
      (catcmpfile2) CATCHPHRASE();                                  1C7M2
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
            ENDCASE;
               BEGIN
               DISABLE (catcmpfile2);
               BUMP errors;
               TERMINATE;  %goes to "ecmp"%
               END;
         CONTINUE;
         END;
END.
%%                                                              1C70
```

```
(cplex) PROCEDURE % get stids of [created] stmnts under file
driving stmnt %                                                    1C8
   % FORMALS %
      ;
   % LOCALS %
      LOCAL TEXT POINTER tp1, tp2, fp1, fp2, fp3, fp4;
      LOCAL STRING flname[200];
      LOCAL STRING locstr[300];
   CASE (procstid _ getsub( cfstid)) OF
      = cfstid:   % must create the statements %
         BEGIN
         % create compile statement next %
            fp1 _ fp2 _ fp3 _ fp4 _ garray[us];
               fp1[1] _ garray[us+1];
               fp2[1] _ garray[ue+1];
               fp3[1] _ garray[fs+1];
               fp4[1] _ garray[fe+1];
               IF FIND fp4 < [";] ^fp4 THEN NULL;
               IF FIND fp4 < [".] ^fp4 THEN NULL;
            *locstr* _ "Compile <";
               IF fp1[1] # fp2[1] THEN *locstr* _ *locstr*, -fp1
               fp2, ",;
               *locstr* _ *locstr*, -fp3 fp4, ",1> Using ",
               *compiler*, " To <";
               IF fp1[1] # fp2[1] THEN *locstr* _ *locstr*, -fp1
               fp2, ",;
               *locstr* _ *locstr*, -fp3 fp4, ".rel,>";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            relstid _ cinssta( cfstid, -1, $tp1, $tp2);
         % create conditional statement next %
            *locstr* _ "Conditionals;";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            constid _ cinssta( cfstid, -1, $tp1, $tp2);
         % create include statement first %
            *locstr* _ "Includes;";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            incstid _ cinssta( cfstid, -1, $tp1, $tp2);
         % create index statement next %
            fp1 _ fp2 _ fp3 _ fp4 _ garray[us];
               fp1[1] _ garray[us+1];
               fp2[1] _ garray[ue+1];
               fp3[1] _ garray[fs+1];
               fp4[1] _ garray[fe+1];
               IF FIND fp4 < [";] ^fp4 THEN NULL;
               IF FIND fp4 < [".] ^fp4 THEN NULL;
            *locstr* _ "<;
               IF fp1[1] # fp2[1] THEN *locstr* _ *locstr*, -fp1
```

```
            fp2, ",";
            *locstr* _ *locstr*, "index-", -fp3 fp4, ",>";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         sysstid _ cinssta( cfstid, -1, $tp1, $tp2);
      % create print history statement next %
         *locstr* _ "Print History";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         prntstid _ cinssta( cfstid, -1, $tp1, $tp2);
      % create update history statement next %
         *locstr* _ "Update History";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         uphstid _ cinssta( cfstid, -1, $tp1, $tp2);
      % create processing history statement next %
         *locstr* _ "Processing History";
         tp1 _ tp2 _ $locstr;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ locstr.L + 1;
         procstid _ cinssta( cfstid, -1, $tp1, $tp2);
      END;
   ENDCASE     % presumably they exist %
      BEGIN
      IF getftl( procstid) THEN REPEAT CASE( cfstid)
      ELSE uphstid _ getsuc( procstid);
      IF getftl( uphstid) THEN REPEAT CASE( cfstid)
      ELSE prntstid _ getsuc( uphstid);
      IF getftl( prntstid) THEN REPEAT CASE( cfstid)
      ELSE sysstid _ getsuc( prntstid);
      IF getftl( sysstid) THEN REPEAT CASE( cfstid)
      ELSE incstid _ getsuc( sysstid);
      IF getftl( incstid) THEN REPEAT CASE( cfstid)
      ELSE constid _ getsuc( incstid);
      IF getftl( constid) THEN REPEAT CASE( cfstid)
      ELSE relstid _ getsuc( constid);
      END;
RETURN;
END.
%%                                                    1C8F
```

```
(cgplex) PROCEDURE % get stid history stmnt under CB/RF/NP
driving stmnt %                                                    1C9
   % FORMALS %
      ;
   % LOCALS %
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING locstr[100];
   CASE (procstid _ getsub( cfstid)) OF
      = cfstid:   % must create the statements %
         BEGIN
         % create processing history statement next %
            *locstr* _ "Processing History";
            tp1 _ tp2 _ $locstr;
               tp1.stastr _ tp2.stastr _ TRUE;
               tp1[1] _ 1;
               tp2[1] _ locstr.L + 1;
            procstid _ cinssta( cfstid, -1, $tp1, $tp2);
         END;
      ENDCASE;    % presumably it exists %
   RETURN;
   END.
   %%                                                              1C9F
```

```
(cpysplex) PROCEDURE % copy plex from sysgd to supersysgd %   1C10
   % FORMALS %
      ;
   % LOCALS %
      LOCAL plsstid;
      LOCAL plestid;
   IF (NOT sprstid) OR (NOT sysorg) THEN RETURN;
   wrtmsg( 0, $": Copying index to sysguide");
   IF (plsstid _ getnxt( sysorg)) # endfil THEN
      BEGIN
      plestid _ getail( plsstid);
      copgrp( sprstid, -1, plsstid, plestid, FALSE);
      BUMP numsysguides;
      END;
   RETURN;
   END.
   %%                                                    1C10H
```

```
(crehis) PROCEDURE % get lib history  & 1st branch stids %     1C11
   % FORMALS %
      ;
   % LOCALS %
      (retvalue) ;                                              1C11B1
      LOCAL TEXT POINTER tp1, tp2;
   IF (lhsstid _ getsub( topstid)) = topstid THEN RETURN( -1);
   tp1 _ lhsstid;   tp1[1] _ 1;
   IF NOT FIND tp1 > [*libhis*] THEN
      BEGIN
      tp1 _ $libhis;   tp1.stastr _ TRUE;   tp1[1] _ 1;
      tp2 _ tp1;   tp2[1] _ libhis.L + 1;
      INVOKE (catcrehis, retcrehis);
      cfstid _ lhsstid := cinssta( topstid, -1, $tp1, $tp2);
      retvalue _ 0;
      END
   ELSE
      IF (cfstid _ getsuc( lhsstid)) = topstid THEN
         retvalue _ - 1
      ELSE retvalue _ 0;
   (retcrehis):                                                 1C11G
   DROP (catcrehis);
   RETURN( retvalue );
   % catchphrases %
      (catcrehis) CATCHPHRASE();                                1C11J1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               retvalue _ 1;
               TERMINATE;
               END;
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                           1C11L
```

```
(dofile) PROCEDURE % process one file %                              1C12
   % FORMALS %
      (cp);        % char count in cfstid after *srcfile* %        1C12A1
   % LOCALS %
      LOCAL fl;   REF fl;
      LOCAL didsomething;
      LOCAL dirnum;
      LOCAL init;
      LOCAL oldcp;
      LOCAL diff;
      LOCAL temp;
      LOCAL savgoutput;
      LOCAL savpgoutput;
      LOCAL TEXT POINTER tp1, tp2, tp3;
      LOCAL STRING l3str[300];
      LOCAL STRING lkstr[100];
      LOCAL STRING messtr[100];
      LOCAL STRING locstr[2000];
      LOCAL STRING stacop[2000];
   % get copy of driving statement for later copying %
      tp1 _ cfstid;   tp1[1] _ 1;
      *stacop* _ tp1 SE( tp1);
   % indicate nothing done intitially %
      didsomething _ 0;
   % get stid for source file and sub-plex %
      IF NOT srcstid _ gmstid( cfstid, cp, TRUE, FALSE ) THEN
         BEGIN
         wrtmsg( 0, $": unable to parse link or load file for
         this branch, branch ignored");
         RETURN;
         END;
      &fl _ flntadr( srcstid.stfile);
      IF fl.fllock AND (NOT fl.flpart) THEN
         BEGIN
         BUMP numlocked;
         !gtfdb( fl.florig, 1B6 + 24B, $R3);
         init _ R3.lkinit;
         dirnum _ R3.lkdirn;
         lockid( $lkstr, dirnum, init);
         IF FIND SE(*lkstr*) ^tp2 [^(] SP 1$ULD ^tp1 THEN
            *lkstr* _ + tp1 tp2;
         *messtr* _
            ": LOCKED BY ", *lkstr*, "; Processing started";
         END
      ELSE *messtr* _ ": Processing started";
      wrtmsg( 0, $messtr);
      cplex();
   % get write date for this file %
      sourcedate _ srcwrd _ getwrtdate( srcstid);
   % get compile parameters %
      INVOKE (catdofile, retdofile);
      relfiledate _ reldate _ getcompile();
      DISABLE(catdofile);  %dropped later%
   % initialize builtin flags %
      % get dates librarian last performed action %
         lpdate _ gtdate( prntstid);
```

```
            lxdate _ gtdate( sysstid);
            iudate _ gtdate( uphstid);
            lidate _ gtdate( incstid);
            lndate _ gtdate( constid);
            lcdate _ gtdate( relstid);
         % get stid and write date for index file %
            IF NOT sysorg _ gmstid( sysstid, 1, FALSE, TRUE) THEN
            inxdate _ 0
            ELSE inxdate _ getwrtdate( sysorg);
         % indicate whether or not we have a pc %
            filepc _ IF fl.flpart THEN 1 ELSE -1;
            nopc _ - filepc;
         % indicate any compile errors %
            cmperr _ 0;
            cmpok _ 1;
         % indicate if source newer than rel %
            newfile _ IF srcwrd > reldate THEN 1 ELSE -1;
            oldfile _ - newfile;
         % indicate not compiled yet %
            filcmp _ -1;
            filncmp _ 1;
      % parse any pre commands (and handle output redirection) %
         savgoutput _ goutput;
         IF NOT preparsecmd( cfstid, oldcp _ cp : cp ) THEN
            BEGIN
            wrtmsg( cfstid, $": unable to parse commands for this
            file, branch ignored");
            RETURN;
            END;
         IF gabort >= 0 THEN RETURN;
         IF gignore >= 0 THEN
            BEGIN
            wrtmsg( 0, $": File ignored");
            RETURN;
            END;
         IF savgoutput # goutput THEN
            BEGIN
            jfntostr( goutput.RH, $13str, 011110B6 .V 1);
            *13str* _ "Changing primary output file to: ", *13str*;
            dismes( 1, $13str);
            !spjfn( 400000B, goutput);
            END;
         IF NOT sprstid THEN gsupersysgd _ -1;
         IF oldcp # cp THEN
            BEGIN
            diff _ oldcp - cp;
            FOR temp _ $garray + 2 UP 2 UNTIL > $garray + 28 DO
               [temp] _ [temp] - diff;
            END;
      % update the file if asked for (maybe) %
         CASE gupdate OF
            < 0: NULL;
            = 0: IF srcwrd = 35M THEN REPEAT CASE( gupdate _ 1);
            > 0: didsomething _ didsomething + upfile();
            ENDCASE;
      % print file, gen & update & print sysgd, count (maybe) %
```

```
      didsomething _ didsomething + prntfile();
   % compile file (maybe) %
      didsomething _ didsomething + cmpfile();
   % process any post commands %
      savpgoutput _ goutput;
      IF NOT postparsecmd( cfstid, cp ) THEN
         BEGIN
         wrtmsg( 0, $": unable to parse post commands for this
         file");
         RETURN;
         END;
      IF savpgoutput # goutput THEN
         BEGIN
         jfntostr( goutput.RH, $13str, 011110B6 .V 1);
         *13str* _ "Changing primary output file to: ", *13str*;
         dismes( 1, $13str);
         !spjfn( 400000B, goutput);
         IF savpgoutput # savgoutput THEN clspjfn(
         savpgoutput.RH);
         END;
      IF gabort >= 0 THEN RETURN;
   % indicate file processed %
      IF lkstr.L THEN *messtr* _ "LOCKED BY ", *lkstr*, "; "
      ELSE *messtr* _ NULL;
      *messtr* _ *messtr*, "Processed by";
      tp1 _ tp2 _ $stacop;
         tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ stacop.L + 1;
         tp3 _ cfstid;   tp3[1] _ 1;
            *locstr* _ tp3 SE( tp3);
         IF (*locstr* # *stacop*) THEN BUMP didsomething;
      IF didsomething THEN
         BEGIN
         cinssta( procstid, -1, $tp1, $tp2);
         rplpdate( cfstid, $messtr);
         END;
   wrtmsg( 0, $": Done processing");
   (retdofile):                                              1C12P
   DROP (catdofile);
   RETURN;
   % catchpharases %
      (catdofile) CATCHPHRASE();                             1C12S1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN
               DISABLE (catdofile);
               wrtmsg( 0,$": Unable to determine compile
               parameters, branch ignored");
               TERMINATE;
               END;
            ENDCASE;
         CONTINUE;
```

```
        END;
    END.
%%                                                          1C120
```

```
(fndqual) PROCEDURE % find first qualifier %                    1C13
   % FORMALS %
      ;
   % LOCALS %
      LOCAL invert;
      LOCAL var1, var2, m;
      LOCAL TEXT POINTER tp1;
   invert _ 2;
   IF FIND *never* $NP THEN invert _ -1
   ELSE IF FIND *default* $NP THEN invert _ 0
   ELSE IF FIND *always* $NP THEN invert _ 1
   ELSE IF FIND *tft* $NP THEN
      invert _ IF userflags[ guindex()] > 0 THEN 1 ELSE -1
   ELSE IF FIND *tff* $NP THEN
      invert _ IF userflags[ guindex()] < 0 THEN 1 ELSE -1
   ELSE IF FIND *tfm* $NP THEN
      invert _ IF userflags[ guindex()] = 0 THEN 1 ELSE -1
   ELSE IF FIND *tfnt* $NP THEN
      invert _ IF userflags[ guindex()] <= 0 THEN 1 ELSE -1
   ELSE IF FIND *tfnf* $NP THEN
      invert _ IF userflags[ guindex()] >= 0 THEN 1 ELSE -1
   ELSE IF FIND *tfnm* $NP THEN
      invert _ IF userflags[ guindex()] # 0 THEN 1 ELSE -1
   ELSE IF FIND *tfe* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 = var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfne* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 # var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfg* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
      );
      invert _ IF var1 > var2 THEN 1 ELSE -1;
      END
   ELSE IF FIND *tfng* $NP THEN
      BEGIN
      var1 _ userflags[ guindex()];
      FIND ^tp1;
      m _ guindex();
      var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
```

```
    );
    invert _ IF var1 <= var2 THEN 1 ELSE -1;
    END
ELSE IF FIND *tf1* $NP THEN
    BEGIN
    var1 _ userflags[ guindex()];
    FIND ^tp1;
    m _ guindex();
    var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
    );
    invert _ IF var1 < var2 THEN 1 ELSE -1;
    END
ELSE IF FIND *tfn1* $NP THEN
    BEGIN
    var1 _ userflags[ guindex()];
    FIND ^tp1;
    m _ guindex();
    var2 _ IF m # 0 THEN userflags[ m] ELSE getfnum( FIND tp1 >
    );
    invert _ IF var1 >= var2 THEN 1 ELSE -1;
    END;
RETURN( IF invert # 2 THEN TRUE ELSE FALSE, invert);
END.
%%                                                          1C130
```

```
(fsuper) PROCEDURE % final processing super sysgd file %      1C14
   % FORMALS %
      ;
   % LOCALS %
      LOCAL temp;
      LOCAL TEXT POINTER tp1;
   IF NOT sprstid THEN RETURN;
   CASE gsupersysgd OF
      < 0:
         BEGIN
         cdelmodfil( sprstid.stfile);
         wrtmsg( -sprstid.stfile, $": Modifications deleted");
         numsysguides _ 0;
         CASE gpsysguide OF
            = 0:
               BEGIN
               tp1 _ topstid;   tp1[1] _ 1;
               IF FIND tp1 > [$"Printed by"] THEN temp _ gtdate(
               topstid)
               ELSE temp _ 0;
               IF (temp = 0) OR (temp < getwrtdate(sprstid)) THEN
                  REPEAT CASE(1);
               END;
            > 0:
               BEGIN
               oqsysgd( sprstid, FALSE);
               rplpdate( topstid, $"Sysguide Printed by");
               END;
            ENDCASE;
         END;
      = 0: IF newsysgdcnt = 0 THEN REPEAT CASE( -1) ELSE REPEAT
      CASE( 1);
      > 0:
         BEGIN
         upsysgd( sprstid);
         IF gpsysguide >= 0 THEN
            BEGIN
            oqsysgd( sprstid, FALSE);
            rplpdate( topstid, $"Sysguide Created & Printed by");
            END
         ELSE rplpdate( topstid, $"Sysguide Created by");
         RETURN;
         END;
      ENDCASE;
   IF gpsysguide > 0 THEN oqsysgd( sprstid, FALSE);
   RETURN;
   END.
   %%                                                         1C14H
```

```
(getcompile) PROCEDURE % get rel filename and date and compiler
name %                                                          1C15
   % FORMALS %
     ;
   % LOCALS %
     LOCAL jfn;
     LOCAL etirwdate;
     LOCAL TEXT POINTER tp1, tp2;
   % parse address link first %
     tp1 _ relstid;   tp1[1] _ 1;
     lnkprs( $tp1, $aarray);
   % now parse the compile link and set text pointer to it %
     tp1 _ aarray[le];   tp1[1] _ aarray[le+1];
     lnkprs( $tp1, $carray);
     cmpptr _ carray[ls];
     cmpptr[1] _ carray[ls + 1];
   % now get rel file name and set text pointer to it %
     tp1 _ carray[le];   tp1[1] _ carray[le+1];
     lnkprs( $tp1, $carray);
     relptr _ carray[ls];
     relptr[1] _ carray[ls + 1];
     lnbfls( $relptr, 0, $relname);
     *relname* _ *relname*, 0;  %need it in TENEX format for
     JSYS%
     BUMP DOWN relname.L;
   % see if rel file exists %
     IF NOT SKIP !gtjfn( 1B11 .V 1B6, 18M6 .V ($relname + 1))
     THEN RETURN( 0);
     jfn _ R1;
   % it does so get its etirwdate and release jfn %
     !gtfdb( jfn, 1B6 .V 14B, $etirwdate);
     IF NOT SKIP !rljfn( jfn) THEN NULL;
   RETURN( etirwdate);
   END.
   %%                                                           1C15J
```

```
(getfnum) PROCEDURE % read number from command %                    1C16
   % FORMALS %
      ;
   % LOCALS %
      LOCAL minus;
      LOCAL val;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING locstr[50];
   minus _ FALSE;
   IF FIND '- $NP THEN minus _ TRUE;
   IF FIND ^tp1 1$D ^tp2 $NP THEN
      BEGIN
      *locstr* _ tp1 tp2;
      val _ VALUE($locstr);
      IF minus THEN val _ - val;
      END
   ELSE val _ 0;
   RETURN( val);
   END.
   %%                                                              1C16I
```

```
(getwrtdate) PROCEDURE % get write date for passed stid %      1C17
   % FORMALS %
      ( flstid);  % stid of concerned file %                  1C17A1
   % LOCALS %
      LOCAL etirwdate;
      LOCAL fl; REF fl;
   &fl _ flntadr( flstid.stfile);
   IF fl.flpart THEN etirwdate _ 35M
   ELSE !gtfdb( fl.florig, 1B6 .V 14B, $etirwdate);
   RETURN( etirwdate);
   END.
   %%                                                         1C17H
```

```
(gmstid) PROCEDURE % get stid of origin statement for link at
passed stid %                                                    1C18
   % FORMALS %
      ( flstid,    % stid of statement containing link %
      cp,          % start position for link search %
      mode,              % TRUE: copy parsed link array to
      global area %
      crefl);      % if file doesn't exist: TRUE: create; FALSE:
      error %
   % LOCALS %
      LOCAL rhostn;
      (retvalue);                                                1C18B2
      (abortsw) _ FALSE;                                         1C18B3
      LOCAL parray[40];
      LOCAL TEXT POINTER tp1, tp2, tp4, rp1;
      LOCAL STRING locstr[100];
      LOCAL STRING filstr[200];
   % parse link thereby finding its delimeters %
      tp1 _ tp2 _ tp4 _ rp1 _ flstid;
         tp1[1] _ tp4[1] _ cp;
      INVOKE (catgmstid, badretgmstid);  %if "lnkprs" does ABORT%
      lnkprs( $tp1, $parray);
      DROP (catgmstid);
      IF mode THEN FOR mode _ 0 UP UNTIL = 40 DO garray[mode] _
      parray[mode];
      tp1[1] _ parray[ls+1];
      tp2[1] _ rp1[1] _ parray[le+1];
   INVOKE (catgmstid, gmscre);  %if "caddexp" does ABORT%
   caddexp( $tp1, $tp2, lda(), $tp4);
   (gmscre):                                                     1C18F
   DROP (catgmstid);
   IF abortsw THEN
      BEGIN  %ABORT occured%
      IF inptrf THEN
         err($"User terminated processing");  %ABORT from ^O%
      IF crefl THEN  %ABORT in "caddexp"%
         BEGIN  % file does not exist -- create file%
         INVOKE (catgmstid, badretgmstid);  %in case create
         fails%
         rhostn _ lnbfls( 0, $parray, $filstr);
         tp4 _ ccrefil( rhostn, $filstr);
         DROP (catgmstid);
         tp4.stpsid _ origin;
         retvalue _ tp4;  %set return value%
         END
      ELSE retvalue _ FALSE;  %file does not exist -- don't
      create%
      END
   ELSE
      BEGIN  %file exists%
      tp4.stpsid _ origin;
      retvalue _ tp4;  %set return value%
      END;
   RETURN( retvalue, rp1[1]);
   (badretgmstid):  %here from catchphrase%                      1C18K
      IF inptrf THEN err($"User terminated processing");
```

```
      RETURN( FALSE);
   %catchphrases%
      (catgmstid) CATCHPHRASE();                                    1C18L1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               BEGIN  %file exists%
               abortsw _ TRUE;
               TERMINATE;
                  %This catchphrase is INVOKE'd and DROP'd
                  several times.  For the first INVOKE,  the
                  TERMINATE goes to 'retgmstid', for the second
                  INVOKE,  the TERMINATE goes to 'gmscre' and for
                  the third INVOKE the TERMINATE goes to
                  'retgmstid' %
               END;
            ENDCASE;
         CONTINUE;
         END;
      (catgmstid1) CATCHPHRASE();                                   1C18L2
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               TERMINATE;
                  %goes to 'retgmstid' for first INVOKE and to
                  'gmscre'  for second INVOKE %
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                               1C18N
```

```
(guindex) PROCEDURE % get index for flag array %                    1C19
   % FORMALS %
      ;
   % LOCALS %
      LOCAL n, dflag, tjfn, tdate;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING nstring[200];
   IF FIND *uflag* ^tp1 1$D ^tp2 $PT $NP THEN
      BEGIN
      *nstring* _ tp1 tp2;
      n _ VALUE( $nstring );
      IF n NOT IN [1, nusrflgs] THEN n _ 0;
      RETURN( n);
      END;
   FIND ^tp1 $PT ^tp2 $NP;
   *nstring* _ tp1 tp2;
   FOR n _ 0 UP UNTIL >= $nbnflgs DO
      IF *nstring* = *[bvarstr[2*n]]* THEN
         RETURN( bvarstr[(2*n)+1] - $userflags);
   dflag _ 0;
   IF *nstring* = *writedate* THEN dflag _ 14B
   ELSE IF *nstring* = *readdate* THEN dflag _ 15B
   ELSE IF *nstring* = *createdate* THEN dflag _ 13B;
   IF dflag > 0 THEN
      BEGIN
      FIND ^tp1 $PT ^tp2 $NP;
      *nstring* _ tp1 tp2, 0;
      dindex _ dindex .X 1;
      n _ $dates + dindex - $userflags;
      userflags[n] _ 0;
      IF SKIP !gtjfn( 10000186, 18M6 .V ($nstring + 1) ) THEN
         BEGIN
         tjfn _ R1;
         !gtfdb( tjfn, 186 .V dflag, $tdate);
         userflags[n] _ tdate;
         IF NOT SKIP !rljfn( tjfn) THEN NULL;
         RETURN( n);
         END;
      END;
   RETURN( 0 );
   END.
   %%                                                               1C19N
```

```
(initprnt) PROCEDURE % initialize printing file %                    1C20
   % FORMALS %
      ;
   % LOCALS %
      LOCAL fl, fjfn;
      LOCAL jfn;
      LOCAL page;
      REF fl;
   INVOKE (catinitprnt, badretinitprnt);
   &fl _ flntadr( sprstid.stfile);
   IF (fjfn _ fl.florig) = 0 THEN RETURN( FALSE);
   jfntostr( fjfn, $qpfilename, 001000B6);
   *qpfilename* _ *qpfilename*, ".LIB-LIST", 0;
   BUMP DOWN qpfilename.L;
   DROP (catinitprnt);
   IF NOT SKIP !gtjfn(4B11 .V 1B6, 18M6 .V ($qpfilename + 1))
   THEN
      RETURN( FALSE);
   jfn _ R1;
   IF NOT SKIP !openf( jfn, 7B10 .V3B5) THEN
      BEGIN
      IF NOT SKIP !rljfn( jfn) THEN NULL;
      RETURN( FALSE);
      END;
   page _ 0;    page.LH _ jfn;
   LOOP
      BEGIN
      IF NOT SKIP !ffufp( page) THEN EXIT LOOP;
      page _ R1;
      !pmap( -1, page, 0);
      BUMP page;
      END;
   IF NOT SKIP !sfptr( jfn, 0) THEN NULL;
   IF NOT SKIP !closf( jfn) THEN RETURN( FALSE);
   RETURN( TRUE);
   (badretinitprnt):  %here from catchphrase%                        1C20F
      DROP (catinitprnt);
      RETURN (FALSE);
   % catchphrases %
      (catinitprnt) CATCHPHRASE();                                   1C20S1
         BEGIN
         CASE SIGNALTYPE OF
            = notetype : NULL;
            = helptype : NULL;
            = aborttype :
               TERMINATE;  %GOES TO "BADRETINITPRNT" %
            ENDCASE;
         CONTINUE;
         END;
   END.
   %%                                                                1C20U
```

```
(isysgd) PROCEDURE % initialize a sysgd file %                    1C21
   % FORMALS %
      ( flstid);  % stid of origin of a sysgd file %             1C21A1
   % LOCALS %
      LOCAL plstid;
      LOCAL plestid;
   IF NOT flstid THEN RETURN;
   % update it first %
      upsysgd( flstid);
   % make sure plex 1 exists %
      plstid _ ccopsta( flstid, -1, flstid, FALSE, 0);
   % now delete plex 1 %
      plestid _ getail( plstid);
      cdelgro( plstid, plestid, 0, 0);
   RETURN;
   END.
   %%                                                            1C21I
```

```
(makexarg)   % CL: ;  make argument for an x-routine %
PROCEDURE (argtype, argvar REF, element1, element2, element3,
element4, element5, element6, element7, element8, element9,
element10);                                                    1C22
   % Procedure description
      FUNCTION
         Create an argument for an x-routine in the format that
         comes out of the argument conversion routine
      ARGUMENTS
         argtype:  type of argument
            value: argxcw -- command word
                   argxvs -- viewspec
         argvar:  adr. of variable for converted argument
            command word -- this is a list
            viewspec -- this is a 4 word array
         element1, element2, ..., element 10:  element value,
         dependent on argtype value
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (vsblk) REF;  %adr. vs block%                            1C2281
      (vs) LIST [1]; %list for vs arg%                         1C2282
   % check argument type and fill argument list %
      CASE argtype OF
         = argxcw:  %command word%
            BEGIN
            IF element1 THEN
               #argvar# _ element1  %token command value%
            ELSE #argvar# _ NULL;
            IF element2 THEN
               #argvar# !_ *[element2]*  %command word%
            ELSE #argvar# !_ NULL;
            END;
         = argxvs:  %viewspec%
            BEGIN
            #vs# _ cnvvsp (element1, element2);  %convert vs
            string%
            &vsblk _ (ELEM #vs#[1]).RH;  %adr. vs block%
            argvar _ vsblk;
            argvar[1] _ vsblk[1];  %copy into passed array%
            argvar[2] _ vsblk[2];
            argvar[3] _ vsblk[3];
            #vs# _ ;  %null out list%
            END;
         ENDCASE;
   % Return %
      RETURN;
      %%                                                       1C22D2
```

```
    END.

(nxtqual) PROCEDURE % find next qualifier %                          1C23
    % FORMALS %
      ( invert);                                                     1C23A1
    % LOCALS %
      LOCAL opor;
      LOCAL opand;
      LOCAL opxor;
      LOCAL ninvert;
      LOCAL var1;
      LOCAL var2;
    opand _ opor _ opxor _ FALSE;
    IF FIND ". ( "o FS opor / "a FS opand / "x FS opxor) $NP THEN
      IF fndqual( : ninvert) THEN
        BEGIN
        var1 _ CASE invert OF
          < 0: 0;
          = 0: 3;
          ENDCASE 6;
        var2 _ CASE ninvert OF
          < 0: 0;
          = 0: 1;
          ENDCASE 2;
        CASE TRUE OF
          = opor:  invert _  ortable[var1 + var2];
          = opand: invert _ andtable[var1 + var2];
          = opxor: invert _ xortable[var1 + var2];
          ENDCASE;
        RETURN( TRUE, invert);
        END;
    RETURN( FALSE, invert);
    END.
    %%                                                               1C23G
```

```
(oplib) PROCEDURE % print the library branch %                    1C24
   % FORMALS %
     ;
   % LOCALS %
     LOCAL da;   REF da;
     LOCAL savvs;
     LOCAL savvs2;
     LOCAL savus;
     LOCAL savca;
     LOCAL savoqapfg;
     LOCAL savcsp;
   IF gplibbranch < 0 THEN RETURN;
   &da _ lda();
   savcsp _ da.dacsp := topstid;
   savoqapfg _ oqapfg := TRUE;
   savca _ da.dacacode := 0;
   savus _ da.dausqcode := 0;
   savvs2 _ da.davspc2 := 0;
   savvs _ da.davspec := 0;
     da.davspec.vslev _ getlev( topstid) + 2;
     da.davspec.vstrnc _ -1;
     da.davspec.vsindf _ da.davspec.vsnamf _ TRUE;
     da.davspec.vsblkf _ da.davspec.vsbrof _ TRUE;
   opseqf( $gpfilename, &da, FALSE, TRUE, opqpfl);
   da.dacsp _ savcsp;
   oqapfg _ savoqapfg;
   da.dacacode _ savca;
   da.dausqcode _ savus;
   da.davspc2 _ savvs2;
   da.davspec _ savvs;
   RETURN;
   END.
   %%                                                            1C24T
```

```
(oqsysgd) PROCEDURE % print a sysgd / index %                    1C25
   % FORMALS %
      ( flstid,    % origin stid sysgd file %
      mode);       % TRUE: file index; FALSE: super sysgd %
   % LOCALS %
      LOCAL da;    REF da;
      LOCAL savqpcolmax;
      LOCAL savvs;
      LOCAL savvs2;
      LOCAL savus;
      LOCAL savca;
      LOCAL savoqapfg;
      LOCAL savcsp;
   IF mode THEN BUMP numpindices ELSE BUMP numpsysguide;
   wrtmsg( -flstid.stfile, $": Printing");
   &da _ lda();
   savqpcolmax _ uqpcolmax := 80;
   savcsp _ da.dacsp := IF mode THEN getnxt( flstid) ELSE flstid;
   savoqapfg _ oqapfg := TRUE;
   savca _ da.dacacode := 0;
   savus _ da.dausqcode := 0;
   savvs2 _ da.davspc2 := 0;
   savvs _ da.davspec := 0;
      da.davspec.vslev _ da.davspec.vstrnc _ IF mode THEN 1 ELSE
      -1;
      da.davspec.vsindf _ da.davspec.vsnamf _ TRUE;
      IF NOT mode THEN da.davspec.vsblkf _ TRUE;
   IF da.dacsp # endfil THEN opseqf( $qpfilename, &da, FALSE,
   TRUE, opqpfl);
   uqpcolmax _ savqpcolmax;
   da.dacsp _ savcsp;
   oqapfg _ savoqapfg;
   da.dacacode _ savca;
   da.dausqcode _ savus;
   da.davspc2 _ savvs2;
   da.davspec _ savvs;
   wrtmsg( -flstid.stfile, $": Done printing");
   RETURN;
   END.
   %%                                                            1C25X
```

```
(pcmdbra) PROCEDURE % process one command branch statement %  1C26
   % FORMALS %
      ;
   % LOCALS %
      LOCAL sgoutput;
      LOCAL STRING 13str[300];
   dismes( 1, $"Processing command statement");
   cgplex();
   ccopsta( procstid, -1, cfstid, FALSE, FALSE);
   sgoutput _ goutput;
   IF NOT postparsecmd( cfstid, 1) THEN
      BEGIN
      dismes( 1, $"*** Error while processing branch");
      END;
   IF sgoutput # goutput THEN
      BEGIN
      jfntostr( goutput.RH, $13str, 011110B6 .V 1);
      *13str* _ "Changing primary output file to: ", *13str*;
      dismes( 1, $13str);
      !spjfn( 400000B, goutput);
      END;
   rplpdate( cfstid, $"Processed by");
   dismes( 1, $"Processing done");
   RETURN;
   END.
   %%                                                    1C26M
```

```
(postparsecmd) PROCEDURE % parse driving statement post commands
%                                                              1C27
   % FORMALS %
     (sstid,      % stid of statement with commands %
     cp);                  % char position after *srcfile* %
   % RETURNS %
     % returns FALSE on error %
   % LOCALS %
     LOCAL savel;
     LOCAL TEXT POINTER cps, cpe;
     LOCAL STRING locstr[1500];
   cpe _ cps _ sstid;   cps[1] _ cp;   cpe[1] _ 1;
   IF NOT FIND cps > [""<<" / "##"] < 2CH ^cps SE( cps) ^cpe THEN
     RETURN( TRUE);
   *locstr* _ cps cpe;
   savel _ locstr.L;
   IF NOT prepost( $locstr) THEN RETURN( FALSE);
   IF savel # locstr.L THEN ST cps cpe _ *locstr*;
   RETURN( TRUE);
   END.
   %%                                                          1C27L
```

```
(prcfile) PROCEDURE % do file using output quickprint / sequence
generator %                                                    1C28
   % FORMALS %
      ;
   % LOCALS %
      LOCAL da;   REF da;
      LOCAL seqa;   REF seqa;
      LOCAL savvs;
      LOCAL savvs2;
      LOCAL savus;
      LOCAL savca;
      LOCAL savoqapfg;
      LOCAL savcsp;
      LOCAL myvspec;
      LOCAL myvspc2;
   % set up and save da record, viewspecs, etc. %
      &da _ lda();
      savcsp _ da.dacsp := srcstid;
      savoqapfg _ oqapfg := TRUE;
      savca _ da.dacacode := $finder;
      savus _ da.dausqcode := 0;
      savvs2 _ da.davspc2 := myvspc2 _ 0;
      savvs _ da.davspec := 0;
         da.davspec.vslev _ da.davspec.vstrnc _ da.davspec.vscapf
         _ da.davspec.vsindf _ da.davspec.vsnamf _
         da.davspec.vsstnf _ da.davspec.vsstnr _ -1;
      myvspec _ da.davspec;
   IF gprint > 0 THEN opseqf( $qpfilename, &da, FALSE, TRUE,
   opqpfl)
   ELSE
      BEGIN
      &seqa _ openseq( srcstid, seqend( srcstid, myvspec,
      myvspc2), myvspec, myvspc2, 0, $finder);
      WHILE (seqgen( &seqa) # endfil) AND (NOT inptrf) DO NULL;
      closeseq( &seqa);
      END;
   % restore da record, etc. %
      da.dacsp _ savcsp;
      oqapfg _ savoqapfg;
      da.dacacode _ savca;
      da.dausqcode _ savus;
      da.davspc2 _ savvs2;
      da.davspec _ savvs;
   RETURN;
   END.
   %%                                                          1C28I
```

```
(preparsecmd) PROCEDURE % parse driving statement pre commands %
                                                                    1C29
   % FORMALS %
     (sstid,     % stid of statement with commands %
     cp);                % char position after *srcfile* %
   % RETURNS %
     % returns FALSE on error %
   % LOCALS %
     LOCAL savel;
     LOCAL TEXT POINTER cps, cpe;
     LOCAL STRING locstr[1500];
   cpe _ cps _ sstid;   cps[1] _ 1;   cpe[1] _ cp;
   IF NOT FIND cpe < [">>" / "##"] > 2CH ^cpe THEN RETURN( TRUE,
   cp);
   *locstr* _ cps cpe;
   savel _ locstr.L;
   IF NOT prepost( $locstr) THEN RETURN( FALSE);
   IF savel # locstr.L THEN
      BEGIN
      ST cps cpe _ *locstr*;
      cp _ cp - (savel - locstr.L);
      END;
   RETURN( TRUE, cp);
   END.
   %%                                                               1C29L
```

```
(prepost) PROCEDURE % parse pre/post commands %                    1C30
   % FORMALS %
      (astr);     % adr string containing commands %              1C30A1
      REF astr;
   % RETURNS %
      % returns FALSE on error %
   % LOCALS %
      LOCAL permc;
      LOCAL cmd;
      LOCAL rstid;
      LOCAL TEXT POINTER tp1, cp1, cp2, cp3, cp4;
   % point to start of string %
      tp1 _ &astr;
         tp1.stastr _ TRUE;
         tp1[1] _ 1;
   % look at what comes next (permanent or temp commands /
   nothing) %
      FIND tp1;
      LOOP
         BEGIN
         FIND > [(("<<" FS permc / "##" FR permc) FS cmd ^cp2 <
         2CH $NP ^cp1) / ENDCHR FR cmd];
         IF NOT cmd THEN EXIT LOOP;
         IF permc THEN
            IF NOT FIND cp2 > [2$">] ^cp4 < 2CH ^cp3 THEN
               BEGIN
               FIND cp2;
               REPEAT LOOP;
               END
            ELSE NULL
         ELSE
            IF NOT FIND cp2 > [2$"#] ^cp4 < 2CH ^cp3 THEN
               BEGIN
               FIND cp2;
               REPEAT LOOP;
               END
            ELSE NULL;
         rdcmd( $cp2, $cp3);
         IF permc THEN FIND cp4
         ELSE
            BEGIN
            ST cp1 cp4 _ NULL;
            FIND cp1;
            END;
         END;
   RETURN( TRUE);
   END.
   %%                                                             1C30H
```

```
(prntfile) PROCEDURE % print file; gen, update, print sysgd;
count; maybe %                                                    1C31
   % FORMALS %
     ;
   % LOCALS %
     LOCAL todo;
     LOCAL indxdate;
     LOCAL lpxdate;
     LOCAL fl; REF fl;
     LOCAL TEXT POINTER tp2, tp3;
     LOCAL STRING locstr[300];
   % get stid and write date for index file %
     IF NOT sysorg THEN
        BEGIN
        wrtmsg( 0, $": Unable to load index file; Index
        functions aborted");
        gpindex _ gsupersysgd _ gsysgd _ -1;
        indxdate _ 35M;
        END
     ELSE
        BEGIN
        &fl _ flntadr( sysorg.stfile);
        IF fl.fllock AND (NOT fl.flpart) THEN
           BEGIN
           BUMP numlocked;
           wrtmsg( 0, $": Index file locked by someone else;
           Index creation aborted");
           gsysgd _ -1;
           indxdate _ 35M;
           END
        ELSE IF (indxdate _ inxdate) = 35M THEN indxdate _ 0;
        END;
   % see if anything to do %
     todo _ FALSE;
     CASE gprint OF
        < 0: NULL;
        = 0:
           IF srcwrd > gtdate( prntstid) THEN gprint _ todo _ 1
           ELSE gprint _ -1;
        > 0: todo _ 1;
        ENDCASE;
     CASE gconcnt OF
        < 0: NULL;
        = 0:
           IF srcwrd > gtdate( constid) THEN gconcnt _ todo _ 1
           ELSE gconcnt _ -1;
        > 0: todo _ 1;
        ENDCASE;
     CASE ginccnt OF
        < 0: NULL;
        = 0:
           IF srcwrd > gtdate( incstid) THEN ginccnt _ todo _ 1
           ELSE ginccnt _ -1;
        > 0: todo _ 1;
        ENDCASE;
     CASE gsysgd OF
```

```
            < 0: NULL;
            = 0:
               IF (srcwrd > indxdate) OR ((indxdate >= libsrttime)
               AND (indxdate # 35M)) THEN
                  gsysgd _ todo _ 1
               ELSE gsysgd _ -1;
            > 0: todo _ 1;
            ENDCASE;
         IF sysorg AND (gsysgd < 0) THEN
            BEGIN
            IF gsupersysgd >= 0 THEN cpysplex();
            CASE gpindex OF
               = 0:
                  BEGIN
                  tp2 _ sysstid;   tp2[1] _ 1;
                  lpxdate _
                     IF FIND tp2 > ["Printed by"] THEN lxdate ELSE
                     0;
                  CASE lpxdate OF
                     = 0: REPEAT CASE 2( 1);
                     < indxdate: IF indxdate # 35M THEN REPEAT CASE
                     2( 1);
                     ENDCASE;
                  END;
               > 0:
                  BEGIN
                  oqsysgd( sysorg, TRUE);
                  rplpdate( sysstid, $"Printed by");
                  clsfile( sysorg := 0);
                  IF NOT todo THEN RETURN( 1);
                  END;
               ENDCASE;
            END;
         IF NOT todo THEN
            BEGIN
            IF sysorg THEN clsfile( sysorg :=0);
            RETURN( 0);
            END;
      % get stid for index file origin %
         IF sysorg THEN
            IF gsysgd < 0 THEN clsfile( sysorg := 0)
            ELSE
               BEGIN
               isysgd( sysorg);
               filenum _ srcstid.stfile;
               *fname* _ NULL;
               filnam( filenum, $fname);
               FIND SF( *fname* ) ["<"] $NP ^tp2 ["."] < CH ^tp3;
               *fname* _ "<", - tp2 tp3, ".";
               END;
      % initialize counters, etc. %
         condcnt _ incldcnt _ filenum _ infile _ inproc _ inrecord _
         0;
      % tell user what we are about to do %
         *locstr* _ NULL;
         IF gprint > 0 THEN *locstr* _ "Printing";
```

```
      IF gsysgd > 0 THEN
         BEGIN
         IF locstr.L THEN *locstr* _ *locstr*, ", ";
         *locstr* _ *locstr*, "Generating index";
         END;
      IF gconcnt > 0 THEN
         BEGIN
         IF locstr.L THEN *locstr* _ *locstr*, ", ";
         *locstr* _ *locstr*, "Counting CONDITIONALs";
         END;
      IF ginccnt > 0 THEN
         BEGIN
         IF locstr.L THEN *locstr* _ *locstr*, ", ";
         *locstr* _ *locstr*, "Counting INCLUDEs";
         END;
      *locstr* _ ": ", *locstr*;
      wrtmsg( 0, $locstr);
   % use either output quickprint or our own sequence generator %
      prcfile();
      IF gprint > 0 THEN
         BEGIN
         ccopsta( prntstid, -1, prntstid, FALSE, FALSE);
         rplpdate( prntstid, $"Printed by");
         END;
   % update counters in driver file %
      IF gconcnt > 0 THEN upconcnt();
      IF ginccnt > 0 THEN upinccnt();
   % tell user we are done %
      *locstr* _ ": Done ", *locstr*[3 TO locstr.L];
      wrtmsg( 0, $locstr);
   % update, print, copy sysgd file %
      IF sysorg AND (NOT inptrf) THEN
         BEGIN
         upsysgd( sysorg);
         ccopsta( sysstid, -1, sysstid, FALSE, FALSE);
         IF gpindex >= 0 THEN oqsysgd( sysorg, TRUE);
         rplpdate( sysstid, IF gpindex >= 0 THEN $"Created &
         Printed by" ELSE $"Created by");
         BUMP newsysgdcnt;
         IF gsupersysgd >= 0 THEN cpysplex();
         END;
   % update builtin counters %
      IF gsysgd > 0 THEN BUMP numindexes;
      IF gconcnt > 0 THEN BUMP numconditionals;
      IF ginccnt > 0 THEN BUMP numincludes;
      IF gprint > 0 THEN BUMP numprints;
   RETURN( 1);
   END.
   %%                                                    1C310
```

```
(rdcmd) PROCEDURE % interpret one command %                    1C32
   % FORMALS %
      ( cps,       % adr text pointer start of command %
      cpe);        % adr text pointer end of command %
   REF cps, cpe;
   % LOCALS %
      LOCAL n;
      LOCAL m;
      LOCAL var1;
      LOCAL var2;
      LOCAL minus;
      LOCAL jfn;
      LOCAL invert;
      LOCAL minutes;
      LOCAL hours;
      LOCAL seconds;
      LOCAL wfor;
      LOCAL wuntil;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING 12str[20];
      LOCAL STRING locstr[1500];
   FIND cps > $NP ^tp1 cpe < $NP ^tp2;
   *locstr* _ - tp1 tp2;
   FIND SF(*locstr*) ^tp1;
   IF fndqual( : invert) THEN WHILE nxtqual( invert : invert) DO
   NULL
   ELSE invert _ 1;
   FIND > $NP ^tp1;
   IF FIND *csever* THEN
      BEGIN
      gprint _ gcompile _ gsysgd _ gsupersysgd _ ginccnt _
      gconcnt _ gupdate _ gptypescript _ gplibbranch _ gpindex _
      gpsysguide _ invert;
      RETURN;
      END;
   IF FIND *cslist* THEN
      BEGIN
      gprint _ gptypescript _ gplibbranch _ gpindex _ gpsysguide
      _ invert;
      RETURN;
      END;
   IF FIND *cscomp* THEN
      BEGIN
      gcompile _ invert;
      RETURN;
      END;
   IF FIND *cspfil* THEN
      BEGIN
      gprint _ invert;
      RETURN;
      END;
   IF FIND *cssysg* THEN
      BEGIN
      gsupersysgd _ invert;
      RETURN;
      END;
```

```
    IF FIND *csrunf* THEN
       BEGIN
       grunfile _ invert;
       RETURN;
       END;
    IF FIND *csindx* THEN
       BEGIN
       gsysgd _ invert;
       RETURN;
       END;
    IF FIND *csupdt* THEN
       IF invert < 0 THEN
          BEGIN
          gupdate _ -1;
          RETURN;
          END
       ELSE
          BEGIN
          IF (gupdate _ invert) >= 0 THEN
             IF FIND $NP "old" THEN guptype _ 1
             ELSE IF FIND $NP "new" THEN guptype _ 2
             ELSE IF FIND $NP "compact" THEN guptype _ 3;
          RETURN;
          END;
    IF FIND *cscond* THEN
       BEGIN
       gconcnt _ invert;
       RETURN;
       END;
    IF FIND *csincl* THEN
       BEGIN
       ginccnt _ invert;
       RETURN;
       END;
    IF FIND *csptyp* THEN
       BEGIN
       gptypescript _ invert;
       RETURN;
       END;
    IF FIND *csplib* THEN
       BEGIN
       gplibbranch _ invert;
       RETURN;
       END;
    IF FIND *cscopyprinter* THEN
       BEGIN
       gcopyprinterbranch _ invert;
       RETURN;
       END;
    IF FIND *cspind* THEN
       BEGIN
       gpindex _ invert;
       RETURN;
       END;
    IF FIND *cspsys* THEN
       BEGIN
```

```
      gpsysguide _ invert;
      RETURN;
      END;
   %IF FIND *csdtch* THEN
      BEGIN
      gdetach _ invert;
      RETURN;
      END; %
   IF FIND *cscomm* $NP ^tp1 THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         *locstr* _ tp1 SE(*locstr*);
         dismes( 1, $locstr);
         END;
      RETURN;
      END;
   IF FIND *csstat* $NP ^tp1 THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         n _ guindex();
         IF n = 0 THEN RETURN;
         FIND ^tp2;
         *locstr* _ "Status of ", + tp1 tp2, " is: ",
            *[( CASE userflags[n] OF
               < 0: $"FALSE (";
               = 0: $"MAYBE (";
               ENDCASE $"TRUE (" )]*,
            *[( CASE userflags[n] OF
               < 0: $never;
               = 0: $default;
               ENDCASE $always )]*,
            ");
         dismes( 1, $locstr);
         END;
      RETURN;
      END;
   IF FIND *csoctl* $NP ^tp1 THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         n _ guindex();
         IF n = 0 THEN RETURN;
         FIND ^tp2;
         *locstr* _
            "Octal value of ", + tp1 tp2, " is: ",  STRING(
            userflags[n], 8);
         dismes( 1, $locstr);
         END;
      RETURN;
      END;
   IF FIND *csdcml* $NP ^tp1 THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
```

```
            n _ guindex();
            IF n = 0 THEN RETURN;
            FIND ^tp2;
            *locstr* _
               "Decimal value of ", + tp1 tp2, " is: ",  STRING(
               userflags[n]);
            dismes( 1, $locstr);
            END;
         RETURN;
         END;
      IF FIND *csdate* $NP ^tp1 THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            n _ guindex();
            IF n = 0 THEN RETURN;
            FIND ^tp2;
            CASE userflags[n] OF
               = 35M:
                  BEGIN
                  !odtim( 18M6 .V ($12str + 1), -1, 0);
                  12str.L _ 12str.M;
                     n _ 0;
                     WHILE *12str*[n+1] DO BUMP n;
                     12str.L _ n;
                  END;
               = 0: *12str* _ "NEVER";
               ENDCASE
                  BEGIN
                  !odtim( 18M6 .V ($12str + 1), userflags[n], 0);
                  12str.L _ 12str.M;
                     n _ 0;
                     WHILE *12str*[n+1] DO BUMP n;
                     12str.L _ n;
                  END;
            *locstr* _
               "Date value of ", + tp1 tp2, " is: ", *12str*;
            dismes( 1, $locstr);
            END;
         RETURN;
         END;
      IF FIND *csoutp* THEN
         BEGIN
         % This is the command to change the primary output file.
         It should result in a call on the FE.  This FE feature has
         not yet been implemented.  Therefore, the command will be
         ignored and an error message will be displayed to the user.
         %
         *locstr* _ "Feature not implemented: command ignored",
            CR, LF, cps cpe;
         dismes(1, $locstr);
         RETURN;
         END;
      IF FIND *for* $NP THEN
         BEGIN
         IF invert >= 0 THEN
```

```
            BEGIN
            var1 _ CASE userflags[ n _ guindex()] OF
               < 0: 0;
               = 0: 3;
               ENDCASE 6;
            var2 _ CASE userflags[ m _ guindex()] OF
               < 0: 0;
               = 0: 1;
               ENDCASE 2;
            userflags[ n] _ ortable[ var1 + var2];
            END;
         RETURN;
         END;
      IF FIND *fand* $NP THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            var1 _ CASE userflags[ n _ guindex()] OF
               < 0: 0;
               = 0: 3;
               ENDCASE 6;
            var2 _ CASE userflags[ m _ guindex()] OF
               < 0: 0;
               = 0: 1;
               ENDCASE 2;
            userflags[ n] _ andtable[ var1 + var2];
            END;
         RETURN;
         END;
      IF FIND *fxor* $NP THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            var1 _ CASE userflags[ n _ guindex()] OF
               < 0: 0;
               = 0: 3;
               ENDCASE 6;
            var2 _ CASE userflags[ m _ guindex()] OF
               < 0: 0;
               = 0: 1;
               ENDCASE 2;
            userflags[ n] _ xortable[ var1 + var2];
            END;
         RETURN;
         END;
      IF FIND *fget* $NP THEN
         BEGIN
         IF invert >= 0 THEN
            BEGIN
            n _ guindex();
            FIND ^tp1;
            m _ guindex();
            userflags[ n] _
               IF m # 0 THEN userflags[ m]
               ELSE getfnum( FIND tp1 >);
            END;
```

```
      RETURN;
      END;
   IF FIND *fnot* $NP THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         n _ guindex();
         userflags[ n] _ - userflags[ n];
         END;
      RETURN;
      END;
   IF FIND *fsetf* $NP THEN
      BEGIN
      IF invert >= 0 THEN userflags[ guindex()] _ -1;
      RETURN;
      END;
   IF FIND *fsett* $NP THEN
      BEGIN
      IF invert >= 0 THEN userflags[ guindex()] _ 1;
      RETURN;
      END;
   IF FIND *fsetm* $NP THEN
      BEGIN
      IF invert >= 0 THEN userflags[ guindex()] _ 0;
      RETURN;
      END;
   IF FIND *fadd* $NP THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         var1 _ userflags[ n _ guindex()];
         FIND ^tp1;
         IF (m _ guindex()) # 0 THEN var2 _ userflags[ m]
         ELSE var2 _ getfnum( FIND tp1 >);
         userflags[ n] _ var1 + var2;
         END;
      RETURN;
      END;
   IF FIND *fsub* $NP THEN
      BEGIN
      IF invert >= 0 THEN
         BEGIN
         var1 _ userflags[ n _ guindex()];
         FIND ^tp1;
         IF (m _ guindex()) # 0 THEN var2 _ userflags[ m]
         ELSE var2 _ getfnum( FIND tp1 >);
         userflags[ n] _ var1 - var2;
         END;
      RETURN;
      END;
   IF FIND *csrept* THEN
      BEGIN
      grepeat _ invert;
      RETURN;
      END;
   IF FIND *csignr* THEN
```

```
        BEGIN
        IF invert >= 0 THEN gignore _ TRUE ELSE gignore _ -1;
        RETURN;
        END;
    IF FIND *csabrt* THEN
        BEGIN
        IF invert >= 0 THEN gabort _ inptrf _ TRUE
        ELSE gabort _ (inptrf _ 0) - 1;
        RETURN;
        END;
    IF FIND *cslogo* THEN
        BEGIN
        glogout _ invert;
        RETURN;
        END;
        %  wait not implemented in NLS 10
    wfor _ wuntil _ FALSE;
    IF FIND *cswait* $NP ("for" FS wfor / "until" FS wuntil) $NP
    ^tp1 1$D ^tp2 THEN
        IF invert >= 0 THEN
            BEGIN
            *12str* _ tp1 tp2;
            hours _ VALUE( $12str);
            IF FIND ":" ^tp1 2D ^tp2 THEN
                BEGIN
                *12str* _ tp1 tp2;
                minutes _ VALUE( $12str);
                END
            ELSE minutes _ hours := 0;
            gwait _ ((hours * 60) + minutes) * 60;
            IF wfor THEN RETURN;
            IF (hours NOT IN [0, 23]) OR (minutes NOT IN [0, 59])
            THEN
                BEGIN
                gwait _ 0;
                GOTO rerr;
                END;
            !odcnv(0, -1, 0, 0);
                seconds _ R4.RH;
            gwait _ gwait - seconds;
            IF gwait <= 0 THEN gwait _ gwait + (24*60*60);
            RETURN;
            END
        ELSE RETURN;
        %
    IF (n _ guindex()) # 0 THEN
        BEGIN
        userflags[n] _ invert;
        RETURN;
        END;
    (rerr):                                                  1C32AV
    *locstr* _ "Illegal command: ignored", CR, LF, "      ", cps
    cpe;
    dismes( 1, $locstr);
    RETURN;
    END.
```

%%

1C328@

```
(rnfl) PROCEDURE % copy printing file to <*prtdir*> directory %
                                                                    1C33
    % FORMALS %
      ;
    % LOCALS %
      LOCAL ret;
      LOCAL ojfn; % jfn of temporary copy %
      LOCAL njfn; % jfn of printer copy %
      LOCAL fsize[2];
      LOCAL dirnum;
      LOCAL STRING dirstr[50];
      LOCAL STRING locstr[400];
    % see if output should be copied to printer directory %
      IF gcopyprinter < 0 THEN RETURN;
    % initialize %
      ret _ FALSE;
    % get jfn of printing file %
      IF NOT SKIP !gtjfn( 1B11 .V 1B6, 18M6 .V ($qpfilename + 1)
      ) THEN
         GOTO rnflret;
      ojfn _ R1;
    % get name of connected directory %
      !gjinf();
         dirnum _ R2;
      IF SKIP !dirst( 18M6 .V ($dirstr + 1), dirnum) THEN
         BEGIN
         dirstr.L _ dirstr.M;
         dirnum _ 0;
         WHILE *dirstr*[dirnum + 1] DO BUMP dirnum;
         dirstr.L _ dirnum;
         IF dirstr.L THEN *dirstr* _ "<, *dirstr*, ">;
         END;
    % see if anything printed first %
      !gtfdb( ojfn, 2B6 .V 11B, $fsize);
      IF (fsize.RH = 0) AND (fsize[1] = 0) THEN
         BEGIN     % nothing printed %
         *locstr* _
            "No printing output - ", *dirstr*, *qpfilename*,  " :
             ";
         IF SKIP !delf( ojfn) THEN ret _ TRUE;
         IF ret THEN *locstr* _ *locstr*, "Deleted"
         ELSE *locstr* _ *locstr*, "UNDELETEABLE *****";
         GOTO rnfldis;
         END;
    % get jfn fordestination file %
      *locstr* _
         "<, *prtdir*, ">, *initsr*, "-, *qpfilename*,
         ";P777777", 0;
      IF NOT SKIP !gtjfn( 4B11 .V 1B6, 18M6 .V ($locstr + 1) )
      THEN
         BEGIN
         IF NOT SKIP !rljfn( ojfn) THEN NULL;
         GOTO rnflret;
         END;
      njfn _ R1;
    % copy and cleanup %
```

```
    ret _ cflapc( ojfn, njfn);
    IF ret THEN IF NOT SKIP !delnf( ojfn, 2) THEN NULL;
    IF NOT SKIP !rljfn( ojfn) THEN NULL;
    IF NOT SKIP !rljfn( njfn) THEN NULL;
(rnflret):                                                    1C33J
    IF ret THEN
       *locstr* _ *dirstr*, *qpfilename*, " copied to <",
       *prtdir*, ">
    ELSE
       *locstr* _ "UNABLE to copy ", *dirstr*, *qpfilename*, "
       to <", *prtdir*, ">;
(rnfldis):                                                    1C33K
    dismes( 1, $locstr);
RETURN( ret);
END.
%%                                                            1C33N
```

```
(rplpdate) PROCEDURE % replace processing signature in passed
stid %                                                          1C34
   % FORMALS %
      ( flstid,    % statement stid in which to replace processing
      date %
      astr);       % adr string to insert before signature %
   REF astr;
   % LOCALS %
      LOCAL i;
      LOCAL TEXT POINTER tp1, tp2;
      LOCAL STRING dtstr[50];
   tp1 _ flstid;   tp1[1] _ 1;
   LOOP
      IF NOT FIND tp1 > E"(("] ^tp1 E"))"] < 2CH ^tp2 THEN
         BEGIN
         ST tp1 _ SF(tp1) SE(tp1), TAB, "(())";
         REPEAT LOOP;
         END
      ELSE EXIT LOOP;
   !odtim( 18M6 .V ($dtstr + 1), -1, 0);
      dtstr.L _ dtstr.M;
      i _ 1;
      WHILE *dtstr*[i] DO BUMP i;
      dtstr.L _ i - 1;
   ST tp1 tp2 _ SP, *astr*, SP, *initsr*, ": ", *dtstr*, SP;
   RETURN;
   END.
   %%                                                           1C34J
```

```
(runfork) PROCEDURE % process one inferior fork processor %   1C35
   % FORMALS %
      (cp,          % char count in cfstid after *runfil*/*newproc*
      %
      spec);        % TRUE > get result from R1 at fork completion
      %
   % LOCALS %
      LOCAL fjfn;
      LOCAL ijfn;
      LOCAL forknum;
      LOCAL result;
      LOCAL savnm;
      LOCAL savrm;
      LOCAL savgoutput;
      LOCAL savpgoutput;
      LOCAL dtc;
      LOCAL TEXT POINTER tp1, tp2, rf1;
      LOCAL regs[20];
      LOCAL parray[50];
      LOCAL STRING locstr[2000];
      LOCAL STRING l3str[300];
      LOCAL STRING infname[200];
      LOCAL STRING inpname[200];
      LOCAL STRING stacop[2000];
   % get copy of driving statement for later copying %
      tp1 _ cfstid;   tp1[1] _ 1;
      *stacop* _ tp1 SE( tp1);
      cgplex();
   % initialize builtin flags %
      % indicate not run yet %
         infrun _ -1;   infnrun _ 1;
      % indicate run successfully %
         infok _ 1;   infnok _ -1;
   % tell user where we are %
      INVOKE (catrunfork1, retrunfork);
      rf1 _ cfstid;   rf1[1] _ cp;
      lnkprs( $rf1, $parray);
      lnbfls( 0, $parray, $infname);
      rf1[1] _ parray[le+1];
      lnbfls( $rf1, 0, $inpname);
      *locstr* _
         "Fork: ", *infname*, "  Input: ", *inpname*, "
         processing started";
      dismes( 1, $locstr);
      DISABLE (catrunfork1);  %dropped at RETURN%
   % parse any pre commands (and handle output redirection) %
      savgoutput _ goutput;
      IF NOT preparsecmd( cfstid, cp : cp ) THEN
         BEGIN
         dismes( 1, $"Unable to parse pre commands for this
         statement, branch ignored");
         RETURN;
         END;
      IF gabort >= 0 THEN RETURN;
      IF gignore >= 0 THEN
         BEGIN
```

```
            dismes( 1, $"Branch ignored");
            RETURN;
            END;
         IF savgoutput # goutput THEN
            BEGIN
            jfntostr( goutput.RH, $13str, 011110B6 .V 1);
            *13str* _ "Changing primary output file to: ", *13str*;
            dismes( 1, $13str);
            !spjfn( 400000B, goutput);
            END;
      % check to see if we should run %
         IF grunfile < 0 THEN GOTO rfpost;
      % get jfn for inferior fork %
         fjfn _ ijfn _ forknum _ 0;
         INVOKE (catrunfork2, rfcleanup);
         *infname* _ *infname*, 0;
         IF NOT SKIP !gtjfn( 1B11.V 1B6, 18M6 .V ($infname + 1))
         THEN
            BEGIN
            dismes( 1, $"Unable to GTJFN for inferior fork");
            result _ -1;
            GOTO rfcleanup;
            END;
         fjfn _ R1;
      % get jfn for input file %
         *inpname* _ *inpname*, 0;
         IF NOT SKIP !gtjfn( 1B11.V 1B6, 18M6 .V ($inpname + 1))
         THEN
            BEGIN
            dismes( 1, $"Unable to GTJFN for inferior fork input
            file");
            result _ -1;
            GOTO rfcleanup;
            END;
         ijfn _ R1;
         IF NOT SKIP !openf( ijfn, 07B10 .V 2B5) THEN
            BEGIN
            dismes( 1, $"Unable to OPENF for inferior fork input
            file");
            result _ -1;
            GOTO rfcleanup;
            END;
      % save our own state %
         !getnm();
            savnm _ R1;
         !gjinf();
            dtc _ R4;
         IF dtc # -1 THEN
            BEGIN
            !rfmod( 100B);
               savrm _ R2;
            END;
      % create and get into inferior fork %
         IF NOT SKIP !cfork( 2B11) THEN
            BEGIN
            dismes( 1, $"Unable to OPENF for inferior fork input
```

```
            file");
            result _ -1;
            GOTO rfcleanup;
            END;
         forknum _ R1;
         !epcap( forknum, -1, -1);
         !spjfn( forknum, (ijfn * 1B6) .V goutput.RH);
         !stiw( forknum, 0);
         R1.LH _ forknum;   R1.RH _ fjfn;
            !get();
         IF NOT SKIP !rljfn( fjfn := 0) THEN NULL;
      % run it, wait, and get result %
         dismes( 1, $"Starting execution");
         infrun _ 1;   infnrun _ -1;
         BUMP numinferiors;
         !sfrkv( forknum, 0);
         !wfork();
         !rfsts( forknum);
            R1 _ R1 .A 7M6;   R1 _ R1.LH;
            IF R1 # 2 THEN result _ -1
            ELSE
               IF NOT spec THEN result _ 1
               ELSE
                  BEGIN
                  !rfacs( forknum, $regs);
                  result _ regs[1];
                  END;
      (rfcleanup): % cleanup %                                        1C35N
         DROP(catrunfork2);
         dismes( 1, $"Execution completed");
         IF forknum AND ijfn THEN !spjfn( forknum, goutput);
         IF forknum THEN !kfork( forknum := 0);
         IF fjfn THEN IF NOT SKIP !closf( fjfn := 0) THEN NULL;
         IF ijfn THEN IF NOT SKIP !closf( ijfn := 0) THEN NULL;
         !setnm( savnm);
         IF dtc # -1 THEN
            BEGIN
            !sfmod( 100B, savrm);
            !stpar( 100B, savrm);
            END;
         infok _ result;   infnok _ - infok;
         IF result >= 0 THEN BUMP numokinferiors;
      (rfpost): % process any post commands %                         1C35N
         savpgoutput _ goutput;
         IF NOT postparsecmd( cfstid, cp ) THEN
            BEGIN
            dismes( 1, $"Unable to parse post commands for this
            branch");
            RETURN;
            END;
         IF savpgoutput # goutput THEN
            BEGIN
            jfntostr( goutput.RH, $13str, 011110B6 .V 1);
            *13str* _ "Changing primary output file to: ", *13str*;
            dismes( 1, $13str);
            !spjfn( 400000B, goutput);
```

```
        IF savpgoutput # savgoutput THEN clspjfn(
        savpgoutput.RH);
        END;
     IF gabort >= 0 THEN RETURN;
  % indicate branch processed %
     tp1 _ tp2 _ $stacop;
        tp1.stastr _ tp2.stastr _ TRUE;
        tp1[1] _ 1;
        tp2[1] _ stacop.L + 1;
        cinssta( procstid, -1, $tp1, $tp2);
     IF infrun = 1 THEN
        IF infok >= 0 THEN *locstr* _ "Processed and Run
        Successfully by"
        ELSE *locstr* _ "Processed and Run UNSUCCESSFULLY by"
     ELSE *locstr* _ "Processed by";
     rplpdate( cfstid, $locstr);
     IF infrun = 1 THEN  ccopsta( lhsstid, -1, cfstid, 0, 0);
  dismes( 1, $"Done processing");
  (retrunfork):                                                  1C350
  DROP (catrunfork1);  %other catchprase DROP'd earlier%
  RETURN;
  % catchphrases %
     (catrunfork1) CATCHPHRASE();                                1C35T1
        BEGIN
        CASE SIGNALTYPE OF
           = notetype : NULL;
           = helptype : NULL;
           = aborttype :
              BEGIN
              DISABLE (catrunfork1);
              dismes( 1, $"Unable to parse inferior fork
              statement; branch ignored");
              result _ -1;
              TERMINATE;  %goes to 'retrunfork'%;
              END;
           ENDCASE;
        CONTINUE;
        END;
     (catrunfork2) CATCHPHRASE();                                1C35T2
        BEGIN
        CASE SIGNALTYPE OF
           = notetype : NULL;
           = helptype : NULL;
           = aborttype :
              BEGIN
              DISABLE (catrunfork2);
              dismes( 1, $"Unable to run inferior fork");
              TERMINATE;  %goes to 'rfcleanup' %
              END;
           ENDCASE;
        CONTINUE;
        END;
  END.
  %%                                                             1C35V
```

```
(upconcnt) PROCEDURE % update condtional count driving statement
%                                                                    1C36
   % FORMALS %
     ;
   % LOCALS %
     LOCAL TEXT POINTER tp1, tp2;
   ccopsta( constid, -1, constid, FALSE, FALSE);
   tp1 _ constid;   tp1[1] _ 1;
   IF NOT FIND tp1 > $NP $D $NP ^tp2 "Condtionals" THEN
     ST tp1 _ STRING(condcnt), " Conditionals;   (())"
   ELSE ST tp1 tp2 _ STRING( condcnt), SP;
   rplpdate( constid, $"Counted by");
   RETURN;
   END.
   %%                                                                1C36J
```

```
(upfile) PROCEDURE % update file according to gupdate %        1C37
   % FORMALS %
     ;
   % LOCALS %
     LOCAL fl;   REF fl;
     LOCAL type;
     LOCAL fileno;
     LOCAL STRING flname[200];
     LOCAL STRING locstr[300];
   % make sure really needs update %
     &fl _ flntadr( fileno _ srcstid.stfile);
     IF (NOT fl.flpart) AND (guptype # 3) THEN RETURN( 0);
     IF fl.fllock AND (NOT fl.flpart) THEN
        BEGIN
        wrtmsg( 0, $": File cannot be updated - locked by
        someone else");
        RETURN( 0);
        END;
   % tell user which file we are updateing %
     *locstr* _ ": Updating (", *[( CASE guptype OF
        = 1: $"old";
        = 2: $"new";
        = 3: $"compact";
        ENDCASE $"new" )]*, ");
     wrtmsg( 0, $locstr);
   % do it %
     type _ CASE guptype OF
        = 1: oldversion;
        = 2: newversion;
        = 3: upcompact;
        ENDCASE newversion;
     IF type = upcompact THEN clist( 15, fileno, 0);
     cupdfil( fileno, type, 0);
   % tell user we are done %
     BUMP numupdates;
     ccopsta( uphstid, -1, uphstid, FALSE, FALSE);
     rplpdate( uphstid, $"Updated by");
     *locstr* _ ": Done ", *locstr*[3 TO locstr.L];;
     wrtmsg( 0, $locstr);
   RETURN( 1);
   END.
   %%                                                          1C371
```

```
(upinccnt) PROCEDURE % update include count driving statement %
                                                                    1C38
    % FORMALS %
      ;
    % LOCALS %
      LOCAL TEXT POINTER tp1, tp2;
    ccopsta( incstid, -1, incstid, FALSE, FALSE);
    tp1 _ incstid;   tp1[1] _ 1;
    IF NOT FIND tp1 > $NP $D $NP ^tp2 "Includes" THEN
      ST tp1 _ STRING(incldcnt), " Includes;      (())"
    ELSE ST tp1 tp2 _ STRING( incldcnt), SP;
    rplpdate( incstid, $"Counted by");
    RETURN;
    END.
    %%                                                              1C38J
```

```
(upsysgd) PROCEDURE % update a sysgd file %                          1C39
   % FORMALS %
      ( flstid);  % origin stid of sysgd file %                     1C39A1
   % LOCALS %
      (da) REF;                                                     1C39B1
      LOCAL fl;   REF fl;
      LOCAL plsstid;
      LOCAL plestid;
   &da _ lda();  %needed by 'sortgrp'%
   &fl _ flntadr( flstid.stfile);
   IF fl.flpart THEN
      BEGIN
      wrtmsg( -flstid.stfile, $": Updating (Compact)");
      IF (plsstid _ getnxt( flstid)) # endfil THEN
         IF (plestid _ getail( plsstid)) # plsstid THEN
            sortgrp( plsstid, plestid, da.dawid);
      clist( 15, flstid.stfile, 0);
      cupdfil( flstid.stfile, upcompact, 0 );
      wrtmsg( -flstid.stfile, $": Done Updating (Compact)");
      END;
   RETURN;
   END.
   %%                                                               1C39H
```

```
(wrtmsg) PROCEDURE % display message on output file %                1C40
   % FORMALS %
      ( flstid,    % 0 / statement stid containing file name / -
      file number %
      astr);       % adr message %
   REF astr;
   % LOCALS %
      LOCAL TEXT POINTER tp1, tp2, tp3, tp4;
      LOCAL parray[40];
      LOCAL STRING flname[200];
      LOCAL STRING locstr[200];
   CASE tp1 _ flstid OF
      > 0:
         BEGIN
         tp1[1] _ 1;
         lnkprs( $tp1, $parray);
         tp1 _ parray[ls];    tp1[1] _ parray[ls+1];
         tp2 _ parray[le];    tp2[1] _ parray[le+1];
         END;
      = 0:
         BEGIN
         tp1 _ garray[ls];    tp1[1] _ garray[ls+1];
         tp2 _ garray[le];    tp2[1] _ garray[le+1];
         END;
      < 0:
         BEGIN
         filnam( -flstid, $flname);
         tp1 _ tp2 _ $flname;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ flname.L + 1;
         IF FIND tp1 > [";] < CH ^tp3 > [",] ^tp4 THEN
            BEGIN
            *flname* _ tp1 tp3, ",, tp4 tp2;
            tp2[1] _ flname.L + 1;
            END;
         END;
     ENDCASE;
   *locstr* _ - tp1 tp2, *astr*;
   dismes( 1, $locstr);
   RETURN;
   END.
   %%                                                                1C40I
```

```
   (nnumon) PROCEDURE (sw );                                          1C41
      LOCAL i, stid, stnv[100];
      LOCAL TEXT POINTER tp1;
      REF sw;
      IF (fchtxt( stid _ sw.swcstid) > 1) OR (inrecord AND FIND
      ['[])
      OR FIND "%%" / ( '% ('+/'-) UL $ULD '% ) THEN
```

```
    BEGIN
    IF NOT (sw.swvspec.vsstnf := TRUE) THEN stvect( stid,
    sw.swsvw);
    sw.swvspec.vsstnr _ TRUE;
    sw.swvspec.vssidf _ FALSE;
    END
ELSE sw.swvspec.vsstnf _ FALSE;
RETURN;
END.
%%                                                    1C41H
```

```
(tinder) PROCEDURE  % content analyzer %                              1C42
    % FORMALS %
      (sw);                                                           1C42A1
    LOCAL
      frtchr,
      da,
      stid,
      i,
      clvl,
      char,
      cmflg;
    LOCAL
      spnam, splnk, spnum;
    LOCAL TEXT POINTER
      tp1, tp2, tp3, tp4, tp5, tp6, tp7, tp8;
    REF
      da, sw;
    spnam _ 15;
    splnk _ 49;
    spnum _ 70;
    % take care of getting numbers on first %
      FIND ^tp1;
      nnumon( &sw);
      FIND tp1 >;
    IF NOT infile THEN
      BEGIN
      tp1[1] _ fchtxt( sw.swcstid);
      IF FIND tp1 > $NP "FILE" THEN infile _ TRUE;
      RETURN( TRUE);
      END;
    IF gconcnt >= 0 THEN
      IF FIND tp1 > '% ('+/'-) UL $ULD '% THEN BUMP condcnt;
    IF ginccnt >= 0 THEN
      IF FIND tp1 > "INCLUDE" 1$NP THEN BUMP incldcnt;
    IF gsysgd < 0 THEN RETURN( TRUE);
    IF inproc THEN
      BEGIN
      IF FIND tp1 > "END" '. THEN inproc _ FALSE;
      RETURN( TRUE);
      END;
    IF inrecord THEN
      BEGIN
      % get tptrs around name %
        IF NOT FIND tp1 ^tp5 ^tp6 > [''[] ^tp7 < CH $NP ^tp2 > CH
        THEN
          RETURN( TRUE);
      cmflg _ FALSE;
      LOOP
        IF FIND ['% / ';] < CH ^tp5 > THEN
          BEGIN
          CASE READC OF
            = '%:
              BEGIN
              FIND ['% / ENDCHR] ^tp6;
              cmflg _ TRUE;
              END;
```

```
                = ";;
                   BEGIN
                   inrecord _ FALSE;
                   EXIT LOOP;
                   END;
                ENDCASE;
             END
          ELSE EXIT LOOP;
       % now get any fdrcomment %
          IF NOT cmflg THEN
             IF FIND tp7 > [";] THEN
                BEGIN
                FIND < CH ^tp5 > CH [";; / ENDCHR] ^tp6;
                END;
       % get field fdrwidth %
          IF NOT FIND tp7 > $NP ^tp7 1$ULD ^tp8 $NP "] THEN FIND
          tp7 ^tp8;
          *fdrwidth* _ - tp7 tp8;
          IF NOT fdrwidth.L THEN *fdrwidth* _ "subfields";
       % build string to insert %
          *fdrstnum* _ NULL;
             fechnm( sw.swsvw, $fdrstnum);
          *fdrstmnt* _ "(, tp1 tp2, ");
             DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
             spnam;
             *fdrstmnt* _ *fdrstmnt*, *fname*, " 0", STRING(
             getsid( tp1)), ">;
             DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
             splnk;
             *fdrstmnt* _ *fdrstmnt*, "FIELD - ", *fdrwidth*;
             DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
             spnum;
             *fdrstmnt* _ *fdrstmnt*, *fdrstnum*;
             IF tp5[1] < tp6[1] THEN *fdrstmnt* _ *fdrstmnt*, CR,
             "   ", tp5 tp6;
       % insert statement in sysgd file %
          tp1 _ tp2 _ $fdrstmnt;
             tp1.stastr _ tp2.stastr _ TRUE;
             tp1[1] _ 1;
             tp2[1] _ fdrstmnt.L + 1;
          cinssta( sysorg, -1, $tp1, $tp2);
       RETURN( TRUE);
       END;
    frtchr _ fchtxt( sw.swcstid);
    IF (frtchr > 1) AND FIND tp1 > ["PROCEDURE"/"COROUTINE"] ^tp6
    < $UL ^tp5 THEN LOOP
       BEGIN
       FIND $NP;
       CASE char _ READC OF
          = "):
             IF NOT FIND $NP ^tp2 $-"( >$NP ^tp1 THEN EXIT LOOP;
          = "%:
             BEGIN
             FIND [";] $NP;
             REPEAT CASE;
             END;
```

```
    ENDCASE EXIT LOOP;
inproc _ TRUE;
% tp5 & tp6 surround PROCEDURE/COROUTINE %
% tp1 & tp2 surround routine name %
% now collect comments and formals %
   *fdrcomment* _ NULL; cmflg _ FALSE;
   *fdrparams* _ NULL;
   FIND tp6 ^tp3 >;
   CASE READC OF
      = ENDCHR:
         BEGIN
         IF (tp3 _ getnxt( tp3) ) = endfil THEN EXIT LOOP;
         tp3[1] _ 1;
         FIND tp3 >;
         REPEAT CASE;
         END;
      = NP: REPEAT CASE;
      = ";:
         IF NOT cmflg THEN
            IF FIND [";] < CH ^tp3 > CH [;" /ENDCHR] ^tp4
            THEN
               *fdrcomment* _ tp3 tp4
            ELSE
               IF FIND tp2 > [";] < CH ^tp3 > CH [;"
               /ENDCHR] ^tp4 THEN
                  *fdrcomment* _ tp3 tp4;
      = ";:
         BEGIN
         IF NOT cmflg THEN *fdrcomment* _ ";;
         CASE char _ READC OF
            = ";: IF NOT cmflg THEN *fdrcomment* _
            *fdrcomment*, ";;
            = ENDCHR:
               BEGIN
               IF NOT cmflg THEN *fdrcomment* _
               *fdrcomment*, SP;
               IF (tp3 _ getnxt(tp3)) = endfil THEN EXIT
               LOOP;
               tp3[1] _ 1;
               FIND tp3 >;
               REPEAT CASE;
               END;
            ENDCASE
               BEGIN
               IF NOT cmflg THEN *fdrcomment* _
               *fdrcomment*, char;
               REPEAT CASE;
               END;
         cmflg _ TRUE;
         REPEAT CASE;
         END;
      = "(:
         BEGIN
         CASE char _ READC OF
            = "): REPEAT CASE 2;
            = ";:
```

```
                    CASE char _ READC OF
                       = "%: REPEAT CASE 2;
                       = ENDCHR:
                          BEGIN
                          IF (tp3 _ getnxt(tp3)) = endfil THEN
                          EXIT LOOP;
                          tp3[1] _ 1;
                          FIND tp3 >;
                          REPEAT CASE;
                          END;
                       ENDCASE REPEAT CASE;
                 = LD:
                    BEGIN
                    IF fdrparams.L THEN *fdrparams* _
                    *fdrparams*, ", ";
                    *fdrparams* _ *fdrparams*, char;
                    CASE char _ READC OF
                       = ",:
                          BEGIN
                          FIND ^tp3;
                          IF FIND SE(*fdrparams*) < $NP ^tp4
                          THEN
                             fdrparams.L _ tp4[1] - 1;
                          FIND tp3 >;
                          REPEAT CASE 2;
                          END;
                       = "):
                          BEGIN
                          FIND ^tp3;
                          IF FIND SE(*fdrparams*) < $NP ^tp4
                          THEN
                             fdrparams.L _ tp4[1] - 1;
                          FIND tp3 >;
                          REPEAT CASE 2 (char);
                          END;
                       = "%:
                          CASE char _ READC OF
                             = "%: REPEAT CASE 2;
                             = ENDCHR:
                                BEGIN
                                IF (tp3 _ getnxt(tp3)) = endfil
                                THEN
                                   EXIT LOOP;
                                tp3[1] _ 1;
                                FIND tp3 >;
                                REPEAT CASE;
                                END;
                             ENDCASE REPEAT CASE;
                       = ENDCHR:
                          BEGIN
                          *fdrparams* _ *fdrparams*, SP;
                          IF (tp3 _ getnxt(tp3)) = endfil THEN
                          EXIT LOOP;
                          tp3[1] _ 1;
                          FIND tp3 >;
                          REPEAT CASE;
```

```
                        END;
                     ENDCASE
                        BEGIN
                        *fdrparams* _ *fdrparams*, char;
                        REPEAT CASE;
                        END;
                  REPEAT CASE;
                  END;
               = ENDCHR:
                  BEGIN
                  IF (tp3 _ getnxt(tp3)) = endfil THEN EXIT
                  LOOP;
                  tp3[1] _ 1;
                  FIND tp3 >;
                  REPEAT CASE;
                  END;
               ENDCASE REPEAT CASE;
            REPEAT CASE;
            END;
         ENDCASE;
      % build string to insert %
         *fdrstnum* _ NULL;
            fechnm( sw.swsvw, $fdrstnum);
         *fdrstmnt* _ "(, tp1 tp2, ");
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            spnam;
            *fdrstmnt* _ *fdrstmnt*, *fname*, " 0", STRING(
            getsid( tp1)), ">;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            splnk;
            *fdrstmnt* _ *fdrstmnt*, tp5 tp6;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            spnum;
            *fdrstmnt* _ *fdrstmnt*, *fdrstnum*;
            IF *fdrcomment* = "% FORMALS %" THEN *fdrcomment* _
            NULL;
            IF fdrparams.L THEN
               *fdrstmnt* _ *fdrstmnt*, CR, "   FORMAL PARAMTERS(
               ", *fdrparams*, ")";
            IF fdrcomment.L THEN *fdrstmnt* _ *fdrstmnt*, CR, "
            ", *fdrcomment*;
      % insert statement in sysgd file %
         tp1 _ tp2 _ $fdrstmnt;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ fdrstmnt.L + 1;
         cinssta( sysorg, -1, $tp1, $tp2);
      RETURN( TRUE);
      END;
   IF (tp1[1] _ frtchr) > 1 THEN LOOP
      BEGIN
      % get tptrs around name %
         IF NOT FIND tp1 ^tp3 < ") $NP ^tp2 $-"( > $NP ^tp1 THEN
         EXIT LOOP;
      % get fdrtype of declaration %
         FIND tp3 > $NP;
```

```
        CASE READC OF
           = ";:
              BEGIN
              *fdrtype* _ "LOCAL";
              FIND ^tp6;
              END;
           = UL:
              BEGIN
              FIND < CH ^tp5  > $UL ^tp6 ^tp8 $NP;
              *fdrtype* _ tp5 tp6;
              CASE READC OF
                 = UL:
                    BEGIN
                    FIND < CH ^tp5  > $UL ^tp6 ^tp8 $NP;
                    *fdrtype* _ *fdrtype*, SP, tp5 tp6;
                    REPEAT CASE;
                    END;
                 = "%:
                    BEGIN
                    FIND [%" / ENDCHR] $NP;
                    REPEAT CASE;
                    END;
                 ENDCASE;
              FIND tp6 > [;" / ENDCHR] ^tp6;
              END;
           = "%:
              BEGIN
              FIND [%" / ENDCHR] $NP;
              REPEAT CASE;
              END;
           ENDCASE
              BEGIN
              *fdrtype* _ "LOCAL";
              FIND ^tp6;
              END;
        IF *fdrtype* = "RECORD" THEN inrecord _ TRUE
        ELSE
           BEGIN
           IF FIND SF(*fdrtype*) "EXTERNAL" $NP ^tp5 THEN
              *fdrtype* _ "EXT ", tp5 SE(*fdrtype*);
           IF FIND SF(*fdrtype*) [;"CONSTANT" / "ADDRESS"] THEN
              IF FIND tp8 > $NP "= $NP ^tp5 [;"] < CH $NP ^tp7
              THEN
                 *fdrtype* _ *fdrtype*, " =", tp5 tp7;
           END;
     % now get any fdrcomment %
        FIND tp6 > ^tp5;
        IF FIND [%"] THEN FIND < CH ^tp5 > CH [%" / ENDCHR]
        ^tp6;
     % build string to insert %
        *fdrstnum* _ NULL;
           fechnm( sw.swsvw, $fdrstnum);
        *fdrstmnt* _ "(, tp1 tp2, ");
           DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
           spnam;
           *fdrstmnt* _ *fdrstmnt*, *fname*, " 0", STRING(
```

```
            getsid( tp1)), ">;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            splnk;
            *fdrstmnt* _ *fdrstmnt*, *fdrtype*;
            DO *fdrstmnt* _ *fdrstmnt*, SP UNTIL fdrstmnt.L >=
            spnum;
            *fdrstmnt* _ *fdrstmnt*, *fdrstnum*;
            IF tp5[1] < tp6[1] THEN *fdrstmnt* _ *fdrstmnt*, CR,
            "   ", tp5 tp6;
      % insert statement in sysgd file %
         tp1 _ tp2 _ $fdrstmnt;
            tp1.stastr _ tp2.stastr _ TRUE;
            tp1[1] _ 1;
            tp2[1] _ fdrstmnt.L + 1;
         cinssta( sysorg, -1, $tp1, $tp2);
      RETURN( TRUE);
      END;
   tp1[1] _ 1;
   IF FIND tp1 > "FINISH" THEN infile _ FALSE;
   RETURN( TRUE );
   END.
   %%                                                    1C42W
```

```
% REMOVE THIS WHEN NEW NLS WITH CORRECTED 'cnvvsp' HAS BEEN MADE
%
   (cnvvsp) %convert viewspec string to viewspec words%
   PROCEDURE( vstring REF %viewspec string%, windowid);        1C43A
      %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 _ aloblk(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
%%                                                          1C45
```