(address)	<nine, format, 02673>	STRING	4C2J
(author)	<nine, format, 04>	STRING	4C2E
(autnames)	<nine, format, 02676>	STRING	4C2H
(bkstr)	<nine, format, 02671>	STRING	4C2M
(cantmsg)	<nine, format, 02734>	CATCHPHRASE	4C3E
(deldir)	<nine, format, 02030>	PROCEDURE	4C6B
(dispforms)	<nine, format, 0809>	PROCEDURE	4C3A
(format)	<nine, format, 04107>	EXT	4C2B
(fstring)	<nine, format, 0280>	PROCEDURE	4C4B
(getdir)	<nine, format, 01871>	PROCEDURE	4C6C
(idconvert)	<nine, format, 01021>	PROCEDURE	4C4A
(jflg)	<nine, format, 01009>	LOCAL	4C2C
(jnum)	<nine, format, 02678>	STRING	4C2F
(journal)	<nine, format, 061>	PROCEDURE	4C3B
(notjournal)	<nine, format, 0932>	PROCEDURE	4C3C
(org)	<nine, format, 02674>	STRING	4C2I
(pxdirs)	<nine, format, 0162>	PROCEDURE	4C4C
(pxstr)	<nine, format, 02672>	STRING	4C2L
(sf)	<nine, format, 0146>	TEXT POINTER	4C2D
(showdir)	<nine, format, 01839>	PROCEDURE	4C6A
(str)	<nine, format, 02675>	STRING	4C2K
(title)	<nine, format, 02677>	STRING	4C2G
(titpg)	<nine, format, 0130>	PROCEDURE	4C4D
(xdeldir)	<nine, format, 02142>	PROCEDURE	4C5D
(xformat)	<nine, format, 05>	PROCEDURE	4C3D
(xresdir)	<nine, format, 02095>	PROCEDURE	4C5C
(xsetdir)	<nine, format, 02383>	PROCEDURE	4C5B

Format subsystem
The Format subsystem provides tools to help you work with Output Processor directives. See one of the menu items below for information about publications and Output Processor directives, or request a copy of the "Output Processor Users' Guide" from FEEDBACK.
% index in <nls,>
% FIRST SEARCHES
general information about how to use publication facilities in NLS
##<publication, how>## 2A
information about how to format with directives
##<publication, format>## 2B
a list of directives grouped by function
##<publication, directive>## 2C
a list of directives grouped alphabetically
##<publication, directive !list>## 2D
commands in the Format subsystem 3
Delete (Directives in) STRUCTURE (at) DESTINATION OK
The Format command "Delete" removes all of the Output Processor directives in the STRUCTURE you specify. 3A
Delimiters alternative: If you previously changed them
If directives in the structure you are working on use delimiters other than a left period and a right semicolon because you had changed the delimiters in a statement preceding the specified structure (e.g., DLD=... DRD= ...), you must also type the "Delimiters" command word before you type OK.

Delete (Directives in) STRUCTURE (at) DESTINATION Delimiters (Left) CONTENT (Right) CONTENT OK
Type in or point to the characters you wish the system to search for as left and right directive delimiters. If the delimiters are changed in the file, the program will understand and search for the new delimiters.
Insert Format (in file at) DESTINATION (using Format number (type 1 for list)) CONTENT
(Title:) CONTENT
(Author Ident(s):) CONTENT
(Journal Number:) CONTENT (Formatting File)
The Format command "Insert" adds directives and any necessary special characters to a file according to a given predesigned format. You must first point to the file. It then lists your choice of formats. Printed samples of all the formats are available through FEEDBACK. You type the number of the desired format. 3B
effects
Unless the file to be formatted is a Journal file, the system asks you to type the title of the document, a list of author idents (which may include organization and group idents), and finally the Journal number (in case you have a preassigned number that you wish to use). To skip any of these fields in TNLS type CA; in DNLS, type <CTRL-N>. The Insert Format command will insert the directives necessary to produce the format you chose. It may insert directives in the origin statement, in every level-one statement, and, in some cases, in the statement one back from every level-one statement. It also creates a title page as the last level-one branch in the file. The title, authors, and Journal number appear in the title page and, in some cases, in the header or footer. 3B1

modifying the formatted file
The program only inserts text in the file. You may edit any of it as you see fit. For example, if one of the authors does not have an ident, you may add his/her name to the title page afterwards, or you may edit the directives in the origin statement to modify the format. (You should feel fairly confident of your understanding of the Output Processor before you attempt to edit the more complex of these formats!) See directives.

Formats designed for COM
Most of the formats were designed for Computer Output to Microfilm (COM). The intent is to make it very easy for users not experienced with the Output Processor to take advantage of COM's capabilities. 3B2

How to have your format and read it too!
To be safe, you should use the "Format" command on files that previously had no directives in them. Since directives can get in the way of online reading, you may want to:

-Update (file) New
-Execute (command in) Format Insert Format
-Output (to) Com or Output (to) Printer
-Delete Modifications

The Delete Modifications command undoes what the Insert Format command did by removing all changes made to the file since the last update. You are then left with a clean file. You must repeat the "Insert Format" command to prepare it for "Output" a second time. See output printer file.

Set Directive (Filter) OK
The Format command "Set Directive" sets your current content-analyzer filter to show only statements that contain Output Processor. You may then use viewspecs i to turn the filter on and viewspec j to turn it off (that is, to show everything). 3C

Delimiters alternative
If directives in the file you are working on use delimiters other than a left period and a right semicolon (because you changed the delimiters in a preceding statement), you must use the command:

Set Directive (Filter) Delimiters (Left) CONTENT (Right) CONTENT OK:

Type in or point to the left and right directive delimiter characters that you wish the program to use. Each new view of the file (a Jump or Print) through this filter begins with the assumption that the directive delimiters are these new characters. Changes in the delimiters by subsequent directives in the view will not keep them from being viewed. 3C1

Reset Directive (Filter) OK
The Format command "Reset Directive" sets your current content-analyzer filter to what it was before the Set Directive (filter) command. This works for only one window at a time in DNLS. 3D

Universal commands (available in all subsystems)
##<nls, universal>## 3E

% source-code for Format .VBS=0; .PBS; .IgRest; <:b> % 4

MAINTENANCE-INDEX BRANCH for FORMAT subsystem

DESCRIPTION

Text descriptions of the tool or subsystem, describing the functions performed by the tool or subsystem.

CHANGES

DATE TIME IDENT <link to change>

Text descriptions of changes made to the tool or subsystem, each change note should include the name of the person making the change, the date, and a reason for the change, as well as the description of the change.

13-JUL-77 KJM (nine,format,idconvert)

Changed to use the author's address instead of the address of his/her organization.

NEEDS AND POSSIBILITIES

GRAMMAR SOURCE <links to the source branches of the grammar>

Links to many files or branches may be placed in substructure. Library branch instructions may surround links.

Naming Conventions

File: <nine, toolname.nls,>

Branch: <grammar>

GRAMMAR COMPILED <links to the compiled files of the grammar>

File Naming Convention: <nine, toolname.GRAM,>

GRAMMAR COMPACTED <links to the compacted files of the grammar>

File Naming Convention: <nine, toolname.CGR,>

PARSE FUNCTIONS CODE SOURCE <links to the source branches of the parse function code>

Naming Conventions

File: <nine, toolname-pfc.nls,>

Branch: <pfc>

PARSE FUNCTIONS CODE COMPILED <links to the compiled files of the parse function code>

File Naming Convention: <sysnine, toolname.PFC>

PARSE FUNCTIONS DATA SOURCE <links to the source branches of the parse function data>

Naming Conventions

File: <nine, toolname-pfd.nls,>

Branch: <pfd>

PARSE FUNCTIONS DATA COMPILED <links to the compiled files of the parse function code>

File Naming Convention: <sysnine, toolname.PFD>

BACKEND SOURCE <links to the source branches of the backend>

These should be divided into machine independent and operating system interface catagories. Links to many files or branches may be placed in substructure. Library branch instructions may surround links.

Naming Conventions

Branch: <be>

File: <directory, filefunction.NLS>

When ever more than one multi-file tool will occupy the same directory, the file naming convention is:

<directory, toolname-filefunction.NLS>

BACKEND COMPILED <link to the runable file of the backend>

For tools this is a SAV file (named .SAV), for subsystems this is a REL file (named .SUBSYS).

File Naming Conventions

```
        <sysnine, toolname.SAV,>
        <sysnine, toolname.SUBSYS,>
   SHARED FILES <link to files shared with other tools>
   USER DOCUMENTATION <links to the user documentation files>
      There are several ways in which user documentation is
      provided, two being output printer files and user guides
      pointed to by help.  TXT is used for sequential files that are
      not in output processor format.
         Naming Conventions
            Files:
               <xhelp, toolname.nls,>
               <nswdoc, toolname-userguide.nls,>
               <nswdoc, toolname.txt,>
               <nswdoc, toolname-userguide.txt,>
               <nswdoc, toolname.print,>
               <nswdoc, toolname-userguide.print,>
            Branches:
               <toolname>
               <userguide>
   MAINTENANCE DOCUMENT <links to documents describing the internal
   procedures and data structures of this tool or subsystem>
      This documentation is intended to be useful to persons
      debugging or upgrading the tool or subsystem.
      Naming Conventions
         File:  <docnine, toolname-MAINTDOC.NLS,>
         Branch:  <MAINTDOC>
   DATASTRUCTURE DOCUMENT
      links to documents describing the interfaces to this tool or
      subsystem, especially any data structures shared with other
      tools or subsystems.
         Naming Conventions
            File:  <docnine, toolname-DATASTRUCDOC.NLS,>
            Branch:  <DATASTRUCDOC>
   RUNFILES SOURCE <links to NLS RUNFILE source branches>
      that are useful in building new versions of this tool or
      subsystem
      Naming Conventions
         File:  <docnine, toolname-RUNFILES.NLS,>
         Branch:  <RUNFILES>
   RUNFILES <links to RUNFILES>
      File Naming Convention:  <relnine, toolname.RUN,>
% grammar using (arcsubsys,cgcml,) to (relnine,format.cml,) %        4B
   FILE cformat
   % COMPILING INSTRUCTIONS %                                        4B2
      INCLUDE <nine, nls-grammar, flags !subsystems>                4B2A
   % DECLARATIONS %                                                 4B3
      INCLUDE <nine, nls-grammar, declarations !universal>          4B3A
      DECLARE FUNCTION % subsystem xroutines %                      4B3B
          dispforms, journal, notjournal, xformat, xsetdir, xresdir,
         xdeldir;
   % COMMON RULES %                                                 4B4
      INCLUDE <nine, nls-grammar, rules !universal>
   %COMMANDS% SUBSYSTEM format KEYWORD "FORMAT"                     4B5
      insert  COMMAND =  "INSERT" "FORMAT" <"in file at">           4B5A
         dest _ DSEL(#"STATEMENT")
         CLEAR <"using Format number (type 1 for list)">
```

```
        [ "LIST" <"formats"> dispforms() ]
        sent _ LSEL(#"NUMBER")
        journal(dest)
        ( IF RESULT NULL
        / IF NOT RESULT (
           CLEAR <"Title:"> param _ LSEL(#"TEXT")
           CLEAR <"Author Ident(s):"> param2 _ LSEL(#"IDENTLIST")
           CLEAR <"Journal Number:"> param3 _ LSEL(#"NUMBER")
           CLEAR <"Formatting File">
           notjournal(param, param2, param3) ) )
        xformat(sent) ;
     set  COMMAND =  "SET" "DIRECTIVE" <"Filter">                    4B5F
        dent _ NULL
        fdelims
        xsetdir(param, param2) ;
     fdelims =                                                       4B5C
        ( "DELIMITERS"
           <"Left"> param _ LSEL(#"CHARACTER"!0!)
           <"Right"> param2 _ LSEL(#"CHARACTER"!0!)
           CONFIRM
        / CONFIRM
           param _ #"."
           param2 _ #";"
        ) ;
     reset  COMMAND =  "RESET" "DIRECTIVE" <"Filter">                4B5D
        CONFIRM
        xresdir() ;
     delete  COMMAND =  "DELETE" <"Directives in"> dest _ structure
                                                                     4B5E
        <"at"> dest _ DSEL(dest)
         fdelims
        xdeldir(dest, param, param2);
     INCLUDE <nine, nls-grammar, commands !universal>                4B5F
     END.                                                            4B5G
  FINISH
```

```
% be source code (arcsubsys, xL10,) (relnine, format.subsys,) %     4C
   FILE lformat
   % globals %                                                      4C2
      % see also under directive recognition xroutines %
      (format) EXTERNAL _ (                                        4C2B
         $"DISPFORMS", $dispforms,
         $"JOURNAL", $journal,
         $"NOTJOURNAL", $notjournal,
         $"XFORMAT", $xformat,
         $"XSETDIR", $xsetdir,
         $"XRESDIR", $xresdir,
         $"XDELDIR", $xdeldir,
         0,0);
      (jflg) ;                                                     4C2C
      (sf) TEXT POINTER;                                           4C2D
      (author) STRING [500];                                       4C2E
      (jnum) STRING [10];                                          4C2F
      (title) STRING [300];                                        4C2G
      (autnames) STRING [500];                                     4C2H
      (org) STRING [100];                                          4C2I
      (address) STRING [300];                                      4C2J
      (str) STRING [1000];                                         4C2K
      (pxstr) STRING [200];                                        4C2L
      (bkstr) STRING [100];                                        4C2M
   % ..........format xroutines.......... %
      (dispforms) PROCEDURE;                                       4C3A
         *lit* _
            EOL, "0:  title page only",
            EOL, "1:  simple printer format",
            EOL, "2:  journal format",
            EOL, "3:  8 pt News Gothic, level one titles",
            EOL, "4:  9 pt Times Roman, level one titles",
            EOL, "5:  10 pt News Gothic, level one titles",
            EOL, "6:  8 pt News Gothic, lev 1 titles, lev 2
            subtitles, right stmnt nums",
            EOL, "7:  9 pt Times Roman, lev 1 titles, lev 2
            subtitles, right stmnt nums",
            EOL, "8:  8 pt News Gothic, level 1 titles, 2 columns;
            you will have to hand", EOL, "    format to balance
            columns at end of each branch",
            EOL, "9:  9 pt Times Roman, level 1 titles, 2 columns;
            you will have to hand", EOL, "    format to balance
            columns at end of each branch",
            EOL, "10: 9 pt Times Roman, indented paragraphs, no
            statement numbers",
            EOL, "11: ARC userguides format",
            EOL, "......... (Format #) ";
         typelit(nocawait, $lit);
         RETURN;
         END.
      (journal) %extract title, author, and journal number if
      journal item%
      PROCEDURE (fptr REF, rtnlist REF);                           4C3B
         LOCAL ldumme REF;
         LOCAL TEXT POINTER j1, j2, j3, j4, aut1, aut2, pg1, pg2,
         se, d;
```

```
      %set sf to origin of file%
         &ldumme _ ELEM #fptr#[tppair];
         sf _ ldumme ;
         sf.stpsid _ origin;
         sf[1] _ 1;
      IF FIND sf > [" ;"] ["HJOURNAL"] ["" "; "] < 3CH ^j2 $D ^j1
      j2 > [".HED="/".H1="] "" ^j3 [" ";] ^j4 _2j4 ["Author(s):
      "] ["/] $NP ^aut1 [";] < CH $NP ^aut2 > [".IGD"] ^d _4d
      [".PES;"] ^se _5se THEN
         BEGIN
         *author* _ aut1 aut2;
         *title* _ j3 j4;
         *jnum* _ j1 j2 ;
         INVOKE(cantmsg);
         IF FIND se [".PEL;"] ^pg1 _5pg1 $NP (".PGN=PGN-1;
         "/".PN=PN-1; ") ^pg2 THEN ST pg1 pg2 _ NULL;
         ST d se _ ".PN=0;" ;
         #rtnlist#[1] _ jflg _ TRUE;
         RETURN;
         END
      ELSE %not a journal file%
         RETURN(jflg _ FALSE);
      #rtnlist#[1] _ jflg _ TRUE;
      RETURN ;
      END.
   (notjournal) PROCEDURE (tptr, aptr, nptr); %put title, author,
   and journal number in strings%                                  4C3C
      getpstring(tptr, $title);
      getpstring(aptr, $author);
      getpstring(nptr, $jnum);
      RETURN ;
      END.
   (xformat) PROCEDURE (fnumptr);                                  4C3D
      LOCAL STRING fnum[10];
      dpset(dspallf, sf, endfile, endfile) ;
      %convert list of author idents to names, org, and address%
         idconvert();
      %get formatting strings %
         getpstring(fnumptr, $fnum);
         fstring($fnum) ;
      % put format directives in origin; leaves sf in origin
      statement%
         INVOKE (cantmsg);
         IF jflg
            THEN FIND sf > [".PN=0;"] ^sf
            ELSE FIND SE(sf) ^sf;
         ST sf sf _ *str* ;
      %put format directives in all level one statements; leaves
      sf at last level one statement%
         pxdirs();
      %insert title page branch at end of file%
         titpg(jflg);
      RETURN;
      END.
   (cantmsg) CATCHPHRASE;                                          4C3E
      CASE SIGNALTYPE OF
```

```
            = aborttype: dismes(1,$"Can't write on file");
         ENDCASE;
   % ..........format support routines.......... %
      (idconvert) PROCEDURE;                                          4C4A
         %declarations and initializations%
            LOCAL count, idfilno ;
            LOCAL TEXT POINTER idptr1, idptr2 ;
            LOCAL STRING idstr[1000], entstr[2000], temp[2000] ;
            org.L _ address.L _ autnames.L _ temp.L _ count _ 0 ;
            idfilno _ open(0,jflname($"identfile")) ;
            INVOKE(lclsfile);
            FIND SF(*author*) $NP ^idptr2;
         LOOP %for each ident in list%
            BEGIN
            IF NOT FIND idptr2 > $NP ^idptr1 1$(LD/'-) ^idptr2 THEN
            EXIT LOOP;
            *idstr* _ idptr1 idptr2 ;
            IF NOT ckident($idstr, $entstr, idfilno) THEN REPEAT
            LOOP;
            IF NOT org.L THEN %fill in org and address%
               BEGIN
               IF orgrptst($entstr,0) THEN
                  *org* _ " " %will be among authors instead%
               ELSE
                  BEGIN
                  getiorg($entstr,$idstr,0,0) ;
                  IF ckident($idstr, $temp, idfilno) THEN
                  getinam($temp,$org,0,0);
                  END;
               laddress ($entstr, $address, 0, 0, idfilno);
               END;
            %add name to autnames%
               getifnf($entstr,$temp);
               *autnames* _ *autnames*, EOL, *temp* ;
               BUMP count ;
               IF count >=5 THEN EXIT LOOP;
            END;
         IF NOT autnames.L THEN *autnames* _ EOL, *author* ;
         DROP(lclsfile);
         close (idfilno) ;
         RETURN;
         (lclsfile) CATCHPHRASE ;                                     4C4A7
            BEGIN
            CASE SIGNALTYPE OF = aborttype:
               IF idfilno THEN close(idfilno);
               ENDCASE;
            CONTINUE;
            END;
         END.
      (fstring) PROCEDURE (fnum REF);  % format directive strings %
                                                                      4C4B
         IF num.L=0 THEN *num* _ '0;
         CASE VALUE(&fnum) OF
            =0:  %blank format (just does title page)%
               BEGIN
               *str* _ NULL ;
```

```
        *pxstr* _ NULL ;
        *bkstr* _ NULL ;
        END;
    =1:  %simple printer format%
        BEGIN
        *str* _ ".SN=0; .LM=-3; .RM=72; .BRM=68; .SNF=72;
        .SNFShow=<=3; ", ".YBS=1,6p; .YBL=0,2p; .F=", "",
        "page .GPN;", "", ";.H1=", "", *title*, "", "; .PN=0;
        .PES; .FP=FR; .PxPShow=1; .PxFShow=1,2; ", ".PxFYD=1;
        .PxFYS=2; .PxFYU=2;";
        *pxstr* _ ".Grab=6;" ;
        *bkstr* _ NULL ;
        END;
    =2:  %journal format%
        BEGIN
        *str* _ ".HJOURNAL=", "", *author*, " .GDT;  ",
        *jnum*, "", "; .H1=", "", *title*, "", "; .IGD=0;
        .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;" ;
        *pxstr* _ NULL ;
        *bkstr* _ NULL ;
        END;
    =3:  %8 pt News Gothic single columns, level one titles
    %
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .HJRM=68,6.875;
        .BRM=65,6.5;",
            " .TM=4,0.75; .DefaultFont=8p,5,Light;
            .PxFontShow=1; .PxFont[1]=12p,6; .HJournal=", "",
            *author*, " .GDT; ", *jnum*, "", ";
            .HJFont=8p,,Light; .YBHJH1=1,0.25; .YFH=3,0.625;
            .H1=", "", *title*, "", "; .H1Font=10p,6,Medium;
            .H1P=J; .FSw=On; .F=", "", "page .GPN;", "",
            "; .FP=FR; .FFont=8p; .SN=Off; .SNFRel=On;
            .SNF=3,0.375; ",
            ".SNFFont=6p,1,Light; .SNFFontShow=All;
            .SNFShow=<=5; ", "  .YBS=1,6p; .YBL=0,2p;
            .PxFShow=1; .PxFYD=1,4p; .PxFYU=2,28p;
            .PxFYS=2,28p; .BP=J;",
            " .ILev=3,0.2; .LM=-6,-0.2; ", ".PxPShow=1;
            .Leading=Off; .SP=C; .PN=0; .PES;";
        *pxstr* _ ".Grab=6,1.0;" ;
        *bkstr* _ NULL ;
        END;
    =4:  %9 pt Times Roman single columns, level one titles
    %
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .HJRM=68,6.875;
        ", ".BRM=65,6.5; .TM=4,0.75; .DefaultFont=9p,6; ",
        ".PxFontShow=1; .PxFont[1]=14p,6; .HJournal=", "",
        *author*, " .GDT; ", *jnum*, "", ";
        .HJFont=8p,5,Light; .YBHJH1=1,0.25; .YFH=3,0.625;
        .H1=", "", *title*, "", "; .H1Font=12p,6,Medium;
        .H1P=J; .FSw=On; .F=", "", "page .GPN;", "", ";
        .FP=FR; .FFont=8p; .SN=Off; .SNFRel=On;
        .SNF=3,0.375;", "  .SNFFont=6p,1,Light;
        .SNFFontShow=All; .SNFShow=<=5;",
```

```
        "  .YBS=1,8p; .YBL=0,2p; .PxFShow=1; .PxFYD=1,4p;",
        " .PxFYU=2,28p; .PxFYS=2,28p; .BP=J;", "
        .ILev=3,0.25; .BLM=-3,-.25;", " .PxPShow=1;
        .Leading=Off; .SP=C; .PN=0; .PES;";
        *pxstr* _ ".Grab=6,1.0;" ;
        *bkstr* _ NULL ;
        END;
    =5:  %10 pt News Gothic single columns, level one titles
    %
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .HJRM=68,6.875;
        ", ".BRM=65,6.5; .TM=4,0.75; .DefaultFont=10p,5; ",
        ".PxFontShow=1; .PxFont[1]=14p,6; .HJournal=", "",
        *author*, " .GDT; ", *jnum*, "", ";
        .HJFont=8p,,Light; .YBHJH1=1,0.25; .YFH=3,0.625;
        .H1=", "", *title*, "", "; .H1Font=12p,6,Medium;
        .H1P=J; .FSw=On; .F=", "", "page .GPN;", "", ";
        .FP=FR; .FFont=8p; .SN=Off; .SNFRel=On;
        .SNF=3,0.375;", "  .SNFFont=6p,1,Light;
        .SNFFontShow=All; .SNFShow=<=5;", " .YBS=1,8p;
        .YBL=0,2p; .PxFShow=1; .PxFYD=1,4p; .PxFYU=2,28p;
        .PxFYS=2,28p; .BP=J;", " .ILev=3,0.25;
        .BLM=-3,-.25;", " .PxPShow=1; .Leading=Off; .SP=C;
        .PN=0; .PES;";
        *pxstr* _ ".Grab=6,1.0;" ;
        *bkstr* _ NULL ;
        END;
    =6:  %8 pt News Gothic single columns, level one titles,
    level two subtitles %
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .BRM=65,6.5; ",
        ".HJRM=68,6.875; .TM=4,0.75; .DefaultFont=8p,5,Light;
        ", ".PxFontShow=1,2; .PxFont[1]=12p,6;
        .PxFont[2]=10p,5; .HJournal=", "", *author*, " .GDT;
        ", *jnum*, "", "; .HJFont=8p,,Light; .YBHJH1=1,0.25;
        .YFH=3,0.625; .H1=", "", *title*, "", ";
        .H1Font=10p,6,Medium; .H1P=J; .FSw=On; .F=", "",
        "page .GPN;", "", "; .FP=FR; .FFont=8p; .SN=Off;
        .SNFRel=On; .SNF=3,0.375; ", " .SNFFont=6p,1,Light;
        .SNFFontShow=All; .SNFShow=<=5;", "  .YBS=1,6p;
        .YBL=0,2p; .PxFShow=1,2; .PxFYD=1,4p; .PxFYU=1,4p;
        .PxFYS=1,4p; .BP=J;", " .ILev=3,0.2; .LM=-6,-0.4;", "
        .PxPShow=1; .Leading=Off; .PN=0; .PES;";
        *pxstr* _ ".Grab=6,1.0;.GYBS=1,24p;" ;
        *bkstr* _ NULL ;
        END;
    =7:  %9 pt Times Roman single columns, level one titles,
    level two subtitles %
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .BRM=65,6.5; ",
        ".HJRM=68,6.875; .TM=4,0.75; ",
        ".DefaultFont=9p,6,Light; .PxFontShow=1,2;
        .PxFont[1]=14p,6; .PxFont[2]=12p,6; .HJournal=", "",
        *author*, " .GDT; ", *jnum*, "", ";
        .HJFont=8p,5,Light; .YBHJH1=1,0.25; .YFH=3,0.625;
        .H1=", "", *title*, "", "; .H1Font=11p,6,Medium;
```

```
        .H1P=J; .FSw=On; .F=", "", "page .GPN;", "", ";
        .FP=FR; .FFont=9p,,Light; .SN=Off; .SNFRel=On;
        .SNF=3,0.375;", " .SNFFont=6p,1,Light;
        .SNFFontShow=All; .SNFShow=<=5;", " .YBS=1,8p;
        .YBL=0,2p; .PxFShow=1,2; .PxFYD=1,4p; .PxFYU=1,4p;
        .PxFYS=1,4p; .BP=J;", " .ILev=3,0.25; .LM=-6,-0.5;",
        " .PxPShow=1; .Leading=Off; .SP=C; .PN=0; .PES;";
        *pxstr* _ ".Grab=6,1.0;.GYBS=1,24p;" ;
        *bkstr* _ NULL ;
        END;
    =8:  %8 pt News Gothic two columns, level one titles%
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .BRM=65,6.5; ",
        ".HJRM=68,6.875; .TM=4,0.75; .DefaultFont=8p,5,Light;
        ", ".PxFontShow=1; .PxFont[1]=14p,5; .HJournal=", "",
        *author*, " .GD; ", *jnum*, "", "; .HJFont=8p,,Light;
        .YBHJH1=1,0.25; .YFH=,0.625; .H1=", "", *title*, "",
        "; .H1Font=10p,6,Medium; .H1P=J; .FSw=On; .F=", "",
        "page .GPN;", "", "; .FP=FR; .FFont=10p,,Light;
        .SN=Off; .SNFRel=On; .SNF=3,0.375;", "
        .SNFFont=6p,1,Light; .SNFFontShow=All;
        .SNFShow=<=5;", "  .YBS=1,8p; .YBL=0,2p; .PxFShow=1;
        .PxFYD=1,4p; .PxFYU=2,28p; .PxFYS=2,28p; .BP=J;", "
        .ILev=3,.2; .LM=-3,-.2;", " .IMax=9,.75; .PxPShow=1;
        .Leading=Off; .XBC=,0.6; .YFC=1,4p; .SP=C; .PN=0;
        .PES;";
        *pxstr* _ ".Grab=6,1.0;.Columns=2;" ;
        *bkstr* _ " .Columns=1;" ;
        END;
    =9:  %9 pt Times Roman two columns, level one titles%
        BEGIN
        *str* _ ".LMBase=4,1.0; .RM=68,6.875; .BRM=65,6.5; ",
        ".HJRM=68,6.875; .TM=4,0.75; .DefaultFont=9p,6; ",
        ".PxFontShow=1; .PxFont[1]=14p,6; .HJournal=", "",
        *author*, " .GDT; ", *jnum*, "", ";
        .HJFont=8p,5,Light; .YBHJH1=1,0.25; .YFH=3,0.625;
        .H1=", "", *title*, "", "; .H1Font=10p,6,Medium;
        .H1P=J; .FSw=On; .F=", "", "page .GPN;", "", ";
        .FP=FR; .FFont=10p,5,Light; .SN=Off; .SNFRel=On;
        .SNF=3,0.375;", "  .SNFFont=6p,1,Light;
        .SNFFontShow=All; .SNFShow=<=5;", "  .YBS=1,6p;
        .YBL=0,2p; .PxFShow=1; .PxFYD=0; .PxFYU=2,28p;
        .PxFYS=2,28p; .BP=J;", " .ILev=3,.2; .LM=-3,-.2;", "
        .IMax=9,.75; .PxPShow=1; .Leading=Off; .XBC=,0.6;
        .YFC=1,4p; .SP=C; .PN=0; .PES;";
        *pxstr* _ ".Grab=6,1.0;.Columns=2;" ;
        *bkstr* _ " .Columns=1;" ;
        END;
    =10:  %9 pt Times Roman indented paragraphs%
        BEGIN
        *str* _ ".LMBase=,1.0; .RM=72,7.0; .HJRM=72,7.0; ",
        ".TM=4,0.75; .BM=56,10.25; .DefaultFont=9p,6,Medium;
        ", ".BP=J; .SN=0; .SNFShow=Off; .HJournal=", "",
        *author*, " .GDT; ", *jnum*, "", ";
        .HJFont=8p,5,Light; .YBHJH1=1,0.25; .YFH=3,0.625;
        .H1=", "", *title*, "", "; .H1Font=10p,6,Medium;
```

```
            .FSw=On; .F=", "", "page .GPN;", "", "; .FP=FR;
            .FFont=8p,5,Light; .YBS=1,6p; .YBL=0,2p;
            .Leading=Off; ", ".IFirst=5,0.3; .ILev=5,0.3;
            .BLM=-5,-0.3; .PN=0; .PES;" ;
            *pxstr* _ ".Grab=5;" ;
            *bkstr* _ NULL ;
            END;
         =11:  %ARC userguides format%
            BEGIN
            *str* _ ".PN=0; .PES; .DefaultFont=10p,5,0; ",
            ".SNFFont=6p,1,1; .PxFontShow=1; .PxFont[1]=14p,5,0;
            .F=", "", "page .GPN;", "", "; .FP=OddR;
            .H1Font=8p,5,0; .H2Font=8p,5,0; .H1=", "", *title*, "
            .Split; ARC ", *jnum*, "  Rev. .GD;", "", "; .H2=",
            "", ".Section;", "", "; .FFont=8p,5,0; .SN=Off;
            .LMBase=0,1.2; .LM=-3,-.25;", " .ILev=3,.25;
            .RM=72,6.5; .BRM=66,6.1; .SNF=72,6.5;",
            " .SNFShow=<=4; .SNFFont=6p,1,Light;
            .SNFFontShow=All;",
            " .YBL=0,2p; .YBS=1,8p; .BP=J; .PxFShow=1; ",
            ".PxFLD=0; .PxFLS=0; .PxFLU=1,24p; .PxPShow=1;
            .Text[Section]=", "", "", "; ;
            *pxstr* _ ".Grab=7;.Text[Section]=", "", "", "; ;
            *bkstr* _ NULL ;
            END;
         =12:  %empty so far%
            BEGIN
            *str* _ " ";
            *pxstr* _ NULL ;
            *bkstr* _ NULL ;
            END;
         ENDCASE err($"Illegal format number") ;
      RETURN;
      END.
   (pxdirs) PROCEDURE; %puts *pxs@<NUL><NUL><NUL><^A>t end of
   every level one st and *bkstr* at end of every st back from
   level ones %                                                    4C4C
      LOCAL TEXT POINTER bk ;
      bk _ 0 ;
      sf _ getsub(sf);
      IF sf.stpsid = origin THEN RETURN;
      IF pxstr.L=0 AND bkstr.L=0 THEN
         BEGIN
         sf _ getail(sf);
         RETURN;
         END;
      LOOP
         BEGIN
         FIND SE(sf) ^sf;
         ST sf sf _ *pxstr* ;
         bk _ getbck(sf) ;
         FIND SE(bk) ^bk ;
         ST bk bk _ *bkstr* ;
         IF getft](sf) THEN EXIT LOOP ;
         sf _ getsuc(sf) ;
         END;
```

```
      RETURN;
      END.
   (titpg) PROCEDURE(jfg);                                          4C 4D
      LOCAL TEXT POINTER ptr1, ptr2 ;
      LOCAL STRING txt[550];
      %insert title page%
         *txt* _ "TITLE PAGE .IgD=0;",
         ".IgText=0;.Post=1;.Trun=All;.IgLS;.H1Sw=0;.H2Sw=0;.H3Sw
         =0;",
         ".H4Sw=0;.SN=0;.SNFShow=Off;.VBS=0;.VBL=0;.Columns=1;.PE
         S;" ;
         FIND SF(*txt*) ^ptr1 SE(*txt*) ^ptr2 ;
         sf _ cinssta(sf, (IF sf.stpsid=origin THEN -1 ELSE 0),
         $ptr1, $ptr2);
         IF jfg THEN
            *txt* _ ".SP=C;.FSw=0;.GYBL=12,2.0;.Font=24p,6,0;",
            *title*, ".GYEL=14,2.4;.GCR;.Font=12p,5;",
            *autnames*, ".VSplit;.GCR;.Size=10p;", *org*,
            ".GYEL=2,0.5;.GCR;.Size=8p;", *address*
         ELSE
            *txt* _ ".SP=C;.FSw=0;.GYBL=12,2.0;.Font=24p,6,0;",
            *title*, ".GYEL=14,2.4;.GCR;.Font=12p,5;",
            *autnames*,
            ".GYEL=13,1.0;.GCR;.GD;.VSplit;.GCR;.Size=10p;",
            *org*, ".GYEL=2,0.5;.GCR;.Size=8p;", *address* ;
         FIND SF(*txt*) ^ptr1 SE(*txt*) ^ptr2 ;
         cinssta(sf, -1, $ptr1, $ptr2);
         RETURN;
         END.
% ..........directive recognition xroutines.......... %
   % DECLARATIONS %
      % Directive lists-- by letter %
         DECLARE lista = (0);
            %current directives%
            %old directives%
         DECLARE aendlist;
         DECLARE listb = (0,
            %current directives%
               "bfont",
               "blm",
               "bm",
               "bp",
               "brm",
               "boldface",
            %old directives%
               "blmx",
               "bmx",
               "bmy",
               "bodyp",
               "brmx");
         DECLARE bendlist;
         DECLARE listc = (0,
            %current directives%
               "cbl",
               "cbs",
               "cel",
```

```
            "ces",
            "cfit",
            "clev",
            "casemode",
            "center",
            "columns",
         %old directives%
            "cas",
            "cen",
            "cmd",
            "cod",
            "columes");
      DECLARE cendlist;
      DECLARE listd = (0,
         %current directives%
            "d",
            "dcase",
            "diabm",
            "dialm",
            "diarm",
            "diaruling",
            "diasid",
            "diatm",
            "dld",
            "dot",
            "dotfont",
            "dotspacing",
            "dotsplit",
            "dotto",
            "drd",
            "dash",
            "defsyn",
            "defaultfont",
         %old directives%
            "dls",
            "dmax",
            "dmin",
            "dpn",
            "dpr",
            "dsh",
            "dsn",
            "dts",
            "defaultfonts",
            "dir");
      DECLARE dendlist;
      DECLARE liste = (0,
         %current directives%
            "evenpage");
         %old directives%
      DECLARE eendlist;
      DECLARE listf = (0,
         %current directives%
            "f",
            "ffont",
            "flm",
            "fp",
```

```
            "frm",
            "fsw",
            "face",
            "font",
         %old directives%
            "flmx",
            "frmx");
      DECLARE fendlist;
      DECLARE listg = (0,
         %current directives%
            "gcr",
            "gd",
            "gdc",
            "gdt",
            "gdtc",
            "gid",
            "glbl",
            "glbs",
            "glel",
            "gles",
            "gn",
            "gntype",
            "gpn",
            "gsig",
            "gsp",
            "gt",
            "gtab",
            "gybl",
            "gybs",
            "gyel",
            "gyes",
            "grab",
         %old directives%
            "gdatm",
            "gdate",
            "gpgnm",
            "graby",
            "grb",
            "grby",
            "gtb",
            "gtime");
      DECLARE gendlist;
      DECLARE listh = (0,
         %current directives%
            "h1",
            "h1font",
            "h1p",
            "h1sw",
            "h2",
            "h2font",
            "h2p",
            "h2sw",
            "h3",
            "h3font",
            "h3p",
            "h3sw",
```

```
            "h4",
            "h4font",
            "h4p",
            "h4sw",
            "hjfont",
            "hjlm",
            "hjrm",
            "hjournal",
            "hlm",
            "hrm",
            "halt",
            "hyphenate",
         %old directives%
            "h",
            "hed",
            "hfont",
            "hj",
            "hjb",
            "hjh",
            "hjl",
            "hjlmx",
            "hjrmx",
            "hjs",
            "hlmx",
            "hln",
            "hlt",
            "hp",
            "hrmx",
            "hsw");
      DECLARE hendlist;
      DECLARE listi = (0,
         %current directives%
            "icr",
            "ifirst",
            "il",
            "ilcr",
            "ilev",
            "imax",
            "iovr",
            "irel",
            "irest",
            "isn",
            "igb",
            "igd",
            "igls",
            "igrest",
            "igs",
            "igtext",
         %old directives%
            "ibr",
            "icrx",
            "ifirstx",
            "igdir",
            "il",
            "ilc",
            "ilcrx",
```

```
            "ilevx",
            "ilo",
            "ilx",
            "imaxx",
            "imx",
            "ind",
            "ins",
            "iovrx",
            "irelx",
            "irestx",
            "irl",
            "irs",
            "isb",
            "isf",
            "isl",
            "isnx",
            "ist");
      DECLARE iendlist;
      DECLARE listj = (0);
         %current directives%
         %old directives%
      DECLARE jendlist;
      DECLARE listk = (0);
         %current directives%
         %old directives%
      DECLARE kendlist;
      DECLARE listl = (0,
         %current directives%
            "lbh1h2",
            "lbh2h3",
            "lbh3h4",
            "lbhjh1",
            "lbl",
            "lbs",
            "lfh",
            "lm",
            "lmbase",
            "lmax",
            "lp",
            "lpf",
            "leading",
            "lev",
            "levclip",
            "levshow",
            "lightface",
         %old directives%
            "lcp",
            "lfl",
            "lms",
            "lmx",
            "lmxbase",
            "lpp",
            "lppn",
            "lsl",
            "lsp",
            "lvl",
```

```
            "lvlshow");
        DECLARE lendlist;
        DECLARE listm = (0,
           %current directives%
              "mcs",
              "mcscn",
              "mcsf",
              "mcsforce",
              "mcsfsw",
              "mcslev",
              "mcsline",
              "mcsonce",
              "mcsrm",
              "mcstext",
              "mcstfont",
              "mcstlpos",
              "mcstpos",
              "mcstrpos",
              "monospace",
           %old directives%
              "maxval",
              "mch",
              "mln",
              "minval");
        DECLARE mendlist;
        DECLARE listn = (0,
           %current directives%
              "names",
              "numdash",
           %old directives%
              "npx",
              "nbl",
              "ndh",
              "ntp",
              "nul");
        DECLARE nendlist;
        DECLARE listo = (0,
           %current directives%
              "oddpage");
           %old directives%
        DECLARE oendlist;
        DECLARE listp = (0,
           %current directives%
              "pbl",
              "pbs",
              "pel",
              "pes",
              "pfit",
              "plev",
              "pn",
              "pntype",
              "pshow",
              "psw",
              "photo",
              "plexnum",
              "post",
```

```
            "pxfld",
            "pxfls",
            "pxflu",
            "pxfshow",
            "pxfyd",
            "pxfys",
            "pxfyu",
            "pxfont",
            "pxfontshow",
            "pxi",
            "pxishow",
            "pxifirst",
            "pxifirstshow",
            "pxirest",
            "pxirestshow",
            "pxix",
            "pxn",
            "pxnfont",
            "pxnfontshow",
            "pxnshow",
            "pxp",
            "pxpshow",
         %old directives%
            "pgn",
            "pgp",
            "pgshow",
            "pln",
            "plo",
            "pno",
            "pnp",
            "pst",
            "pxf");
      DECLARE pendlist;
      DECLARE listq = (0);
         %current directives%
         %old directives%
      DECLARE qendlist;
      DECLARE listr = (0,
         %current directives%
            "rm",
         %old directives%
            "res",
            "rfl",
            "recto",
            "rmx");
      DECLARE rendlist;
      DECLARE lists = (0,
         %current directives%
            "sn",
            "snf",
            "snffont",
            "snffontshow",
            "snfshow",
            "snfont",
            "snfontshow",
            "snfrel",
```

```
            "snftype",
            "snshow",
            "sntype",
            "sp",
            "setl",
            "setr",
            "sigf",
            "sigfont",
            "size",
            "slant",
            "split",
         %old directives%
            "scan",
            "scr",
            "sgf",
            "sigffont",
            "sigfrel",
            "sigfx",
            "sigrel",
            "skp",
            "sna",
            "snb",
            "snfx",
            "ssw");
      DECLARE sendlist;
      DECLARE listt = (0,
         %current directives%
            "tm",
            "tabp",
            "tabstops",
            "tabto",
            "tabs",
            "text",
            "trailing",
            "trun",
         %old directives%
            "tab",
            "tabstopsx",
            "tabtox",
            "tln",
            "tmx",
            "tmy",
            "tsl");
      DECLARE tendlist;
      DECLARE listu = (0,
         %current directives%
            "u0",
            "u1",
            "u2",
            "u3",
            "u4",
            "u5",
            "u6",
            "u7",
            "u8",
            "u9",
```

```
            "underline");
         %old directives%
      DECLARE uendlist;
      DECLARE listv = (0,
         %current directives%
            "v1",
            "v1font",
            "v2",
            "v2font",
            "v3",
            "v3font",
            "vsplit",
         %old directives%
            "v",
            "verso",
            "vfont");
      DECLARE vendlist;
      DECLARE listw = (0,
         %current directives%
            "widowl",
            "wln");
         %old directives%
      DECLARE wendlist;
      DECLARE listx = (0,
         %current directives%
            "x",
            "xbc",
         %old directives%
            "xcl");
      DECLARE xendlist;
      DECLARE listy = (0,
         %current directives%
            "ybh1h2",
            "ybh2h3",
            "ybh3h4",
            "ybhjtm",
            "yb1",
            "ybs",
            "yfc",
            "yfh",
            "ymax",
            "ypf",
         %old directives%
            "ybhjh1");
      DECLARE yendlist;
      DECLARE listz = (0);
         %current directives%
         %old directives%
      DECLARE zendlist;
   DECLARE oldcaf ;
   DECLARE STRING tdld[5], tdrd[5], udld[5], udrd[5] ;
   DECLARE STRING test[11];
   DECLARE TEXT POINTER b, e, t1, t2;
(xsetdir) PROCEDURE (rdlREF, ldl REF);                          4C5B
   LOCAL da REF ;
   *udld* _ *ldl*;
```

```
        *udrd* _ *rdl*;
        &da _ lda() ;
        IF da.dacacode # $showdir THEN
           BEGIN
           oldcaf _ da.dacacode;
           da.dacacode _ $showdir ;
           END;
        dismes (1, $"Use viewspecs i and j to turn on and off") ;
        RETURN;
        END.
     (xresdir) PROCEDURE;                                          4C5C
        LOCAL da REF ;
        &da _ lda() ;
        IF da.dacacode = $showdir THEN
           da.dacacode _ oldcaf;
        RETURN;
        END.
     (xdeldir) PROCEDURE (typestruc REF, ldl REF, rdl REF);        4C5D
        LOCAL da REF, sw REF, vspec, ptr REF ;
        *tdld* _ *ldl*;
        *tdrd* _ *rdl*;
        &ptr _ ELEM#typestruc#[tppair] ;
        CASE ELEM#typestruc#[cwtype] OF
           = 29 %- statement -%:
              deldir (ptr) ;
           = 27 %- group -%, = 28 %- plex -%, = 26 %- branch -%:
              BEGIN
              %set up viewspecs to be followed%
                 &da _ lda();
                 vspec _ da.davspec;
                 vspec.vsbrof _ FALSE;
                 vspec.vsplxf _ FALSE;
              %open sequence work area%
                 &sw _ openseq (ptr, ptr[2], vspec, da.davspc2,
                 da.dausgcod, da.dacacode) ;
              %delete directives in all statements in seq work
              area%
                 WHILE seqgen(&sw) # endfil
                    DO deldir (sw.swstid);
              %close seq work area%
                 closeseq(&sw);
              END;
           ENDCASE err(notyet);
        IF nlmode=fulldisplay THEN alldsp() ;
        RETURN;
        END.
  % ..........directive recognition support routines.......... %
     (showdir) PROCEDURE (sw);                                     4C6A
        REF sw ;
        IF NOT sw.swkflg THEN
           BEGIN
           *tdld* _ *udld* ;
           *tdrd* _ *udrd* ;
           sw.swkflg _ TRUE;
           END;
        LOOP
```

```
            BEGIN
            IF FIND [*tdld*] ^b _b ^t1 THEN
               BEGIN % Left delimiter found; maybe a directive %
               % Is there a right delimiter? %
               IF FIND L $10LD ^t2 [*tdrd*] ^e THEN
                  BEGIN % Right delimiter found; check further. %
                  *test* _ -t1 t2;
                  CCPOS SF(*test*);
                  IF getdir(READC, FALSE) THEN % found one %
                     BEGIN
                     % Directive has  been found and deleted.  Set
                     up pointer before old deleted text and contiue
                     scanning. %
                        RETURN (TRUE) ;
                     END;
                  %not a directive%
                  END;
               %Right delimiter not found or directive not found.
               Advance pointer beyond potential directive and
               continue scanning. (Repeat loop.) %
                  CCPOS t1;
               END
            ELSE RETURN(FALSE); % No more directives in this
            statement %
            END;
         END.
      (deldir) PROCEDURE (stid);                                    4C6B
         FIND SF(stid) ;
         LOOP
            BEGIN
            IF FIND [*tdld*] ^b _b ^t1 THEN
               BEGIN % Left delimiter found; maybe a directive %
               % Is there a right delimiter? %
               IF FIND L $10LD ^t2 [*tdrd*] ^e THEN
                  BEGIN % Right delimiter found; check further. %
                  *test* _ -t1 t2;
                  CCPOS SF(*test*);
                  IF getdir(READC, TRUE) THEN % found one %
                     BEGIN
                     % Directive has been found and deleted.  Set up
                     pointer before old deleted text and continue
                     scanning. %
                        CCPOS b;
                        REPEAT LOOP;
                     END;
                  %not a directive%
                  END;
               %Right delimiter not found or directive not found.
               Advance pointer beyond potential directive and
               continue scanning. (Repeat loop.) %
                  CCPOS t1;
               END
            ELSE RETURN; % No more directives in this statement %
            END;
         END.
      (getdir) PROCEDURE (char,delete);                             4C6C
```

```
% Search for directives.  If found, delete and return true.
 If not found. return false.  Could modify the default
delimiters. %
LOCAL end, list, i;
REF list;
CASE char OF
   ='a:
      BEGIN
      end _ $aendlist;
      &list _ $lista;
      END;
   ='b:
      BEGIN
      end _ $bendlist;
      &list _ $listb;
      END;
   ='c:
      BEGIN
      end _ $cendlist;
      &list _ $listc;
      END;
   ='d:
      BEGIN
      end _ $dendlist;
      &list _ $listd;
      END;
   ='e:
      BEGIN
      end _ $eendlist;
      &list _ $liste;
      END;
   ='f:
      BEGIN
      end _ $fendlist;
      &list _ $listf;
      END;
   ='g:
      BEGIN
      end _ $gendlist;
      &list _ $listg;
      END;
   ='h:
      BEGIN
      end _ $hendlist;
      &list _ $listh;
      END;
   ='i:
      BEGIN
      end _ $iendlist;
      &list _ $listi;
      END;
   ='j:
      BEGIN
      end _ $jendlist;
      &list _ $listj;
      END;
```

```
        ='k:
            BEGIN
            end _ $kendlist;
            &list _ $listk;
            END;
        ='l:
            BEGIN
            end _ $lendlist;
            &list _ $listl;
            END;
        ='m:
            BEGIN
            end _ $mendlist;
            &list _ $listm;
            END;
        ='n:
            BEGIN
            end _ $nendlist;
            &list _ $listn;
            END;
        ='o:
            BEGIN
            end _ $oendlist;
            &list _ $listo;
            END;
        ='p:
            BEGIN
            end _ $pendlist;
            &list _ $listp;
            END;
        ='q:
            BEGIN
            end _ $qendlist;
            &list _ $listq;
            END;
        ='r:
            BEGIN
            end _ $rendlist;
            &list _ $listr;
            END;
        ='s:
            BEGIN
            end _ $sendlist;
            &list _ $lists;
            END;
        ='t:
            BEGIN
            end _ $tendlist;
            &list _ $listt;
            END;
        ='u:
            BEGIN
            end _ $uendlist;
            &list _ $listu;
            END;
        ='v:
```

```
                BEGIN
                end _ $vendlist;
                &list _ $listv;
                END;
            ='w:
                BEGIN
                end _ $wendlist;
                &list _ $listw;
                END;
            ='x:
                BEGIN
                end _ $xendlist;
                &list _ $listx;
                END;
            ='y:
                BEGIN
                end _ $yendlist;
                &list _ $listy;
                END;
            ='z:
                BEGIN
                end _ $zendlist;
                &list _ $listz;
                END;
            ENDCASE RETURN( FALSE );
        FOR i_1 UP UNTIL > (end - &list - 1) DO
            IF *test* = *[list[i]]* THEN
                BEGIN
                CASE char OF
                    ='d:
                        BEGIN
                        IF *test*="dld" THEN
                            IF FIND t2 ['"] ^t1 CH ^t2 THEN
                                *tdld* _ t1 t2;
                        IF  *test*="drd" THEN
                            IF FIND t2 ['"] ^t1 CH ^t2 THEN
                                *tdrd* _ t1 t2;
                        END;
                    ='f, ='h:
                        IF *test*="h1" OR *test*="h2" OR *test*="h3" OR
                        *test*="h4" OR *test*="hjournal" OR *test*="h"
                        OR *test*="hj" OR *test*="hed" OR *test*="f"
                        THEN
                            IF NOT FIND t2 ['" *tdrd*] ^e THEN RETURN(
                            FALSE);
                    ENDCASE;
                IF delete THEN ST b e _ NULL ;
                RETURN( TRUE );
                END;
        %No match found%
        RETURN( FALSE );
        END.
INCLUDE <nine, identsupport, sendmail>
FINISH
```

```
((cdsdwh) RECORD hgt) <nine, graphics, 01704>    FIELD - 16              6C1B13
((comcds) RECORD b7low) <nine, graphics, 01702>  FIELD - 7               6C1B13
(absdif)        <nine, graphics, 06013>          PROCEDURE               6C1V16
(addcalit)      <nine, graphics, 01511>          EXT CONSTANT =13        6C1B12
(adout1)        <nine, graphics, 01347>          EXT                     6C1B3I
(alpham)        <nine, graphics, 01343>          EXT                     6C1B3H
(annotation)    <nine, graphics, 01615>          EXT CONSTANT =14
(arc)           <nine, graphics, 01612>          EXT CONSTANT =10
(arrowhead)     <nine, graphics, 01611>          EXT CONSTANT =9
(ascii0)        <nine, graphics, 01475>          EXT CONSTANT =608       6C1B8A
(aspctr)        <nine, graphics, 01507>          EXT CONSTANT =201542233513B
(atch)          <nine, graphics, 01295>          EXT                     6C1B3A
(attach)        <nine, graphics, 01292>          EXT                     6C1B3A
(atvisible)     <nine, graphics, 01293>          EXT                     6C1B3A
(bedge)         <nine, graphics, 01473>          EXT CONSTANT =95        6C1B7A
(bldtrk)        <nine, graphics, 06272>          PROCEDURE               6C1Z1
(cdsrec)        <nine, graphics, 01703>          RECORD                  6C1B13
(chsize)        <nine, graphics, 01376>          EXT CONSTANT =2         6C1B3L
(circle)        <nine, graphics, 01608>          EXT CONSTANT =6
(clast)         <nine, graphics, 01340>          EXT                     6C1B3H
(comch)         <nine, graphics, 01356>          EXT                     6C1B3I
(comeof)        <nine, graphics, 01351>          EXT                     6C1B3I
(comfig)        <nine, graphics, 01350>          EXT                     6C1B3I
(comfnt)        <nine, graphics, 01354>          EXT                     6C1B3I
(comjfn)        <nine, graphics, 01361>          EXT                     6C1B3I
(comin)         <nine, graphics, 01352>          EXT                     6C1B3I
(comisg)        <nine, graphics, 01348>          EXT                     6C1B3I
(comnui)        <nine, graphics, 01353>          EXT                     6C1B3I
(comp80)        <nine, graphics, 01337>          EXT                     6C1B3F
(compag)        <nine, graphics, 01349>          EXT                     6C1B3I
(comtyp)        <nine, graphics, 01355>          EXT                     6C1B3I
(comxlfb)       <nine, graphics, 01357>          EXT                     6C1B3I
(comxrfb)       <nine, graphics, 01358>          EXT                     6C1B3I
(comybfb)       <nine, graphics, 01360>          EXT                     6C1B3I
(comytfb)       <nine, graphics, 01359>          EXT                     6C1B3I
(devset)        <nine, graphics, 04280>          PROCEDURE               6C1J6
(dfltah)        <nine, graphics, 01314>          EXT STRING              6C1B3D
(dfltaw)        <nine, graphics, 01315>          EXT STRING              6C1B3D
(dfltb)         <nine, graphics, 01321>          EXT                     6C1B3D
(dfltcf)        <nine, graphics, 01324>          EXT                     6C1B3D
(dfltcs)        <nine, graphics, 01313>          EXT STRING              6C1B3D
(dfltg)         <nine, graphics, 01318>          EXT                     6C1B3D
(dfltj)         <nine, graphics, 01320>          EXT                     6C1B3D
(dfltls)        <nine, graphics, 01323>          EXT                     6C1B3D
(dfltlt)        <nine, graphics, 01322>          EXT                     6C1B3D
(dfltr)         <nine, graphics, 01316>          EXT                     6C1B3D
(diamond)       <nine, graphics, 01609>          EXT CONSTANT =7
(earc)          <nine, graphics, 01614>          EXT CONSTANT =12
(east)          <nine, graphics, 01617>          EXT CONSTANT =0
(ellipse)       <nine, graphics, 01613>          EXT CONSTANT =11
(fabs)          <nine, graphics, 06009>          PROCEDURE               6C1V15
(fadd)          <nine, graphics, 05845>          PROCEDURE               6C1V2
(fbaddlit)      <nine, graphics, 01510>          EXT CONSTANT =12        6C1B12
(fbctl)         <nine, graphics, 06344>          PROCEDURE               6C1Z5
(fbendlit)      <nine, graphics, 01513>          EXT CONSTANT =15        6C1B12
(fdiv)          <nine, graphics, 05866>          PROCEDURE               6C1V5
```

(fdrad)	<nine, graphics, 05897>	PROCEDURE	6C1V8
(ffour)	<nine, graphics, 01416>	EXT CONSTANT =2034000000000B	
(fhalf)	<nine, graphics, 01413>	EXT CONSTANT =2004000000000B	
(figure)	<nine, graphics, 01604>	EXT CONSTANT =1	
(fix)	<nine, graphics, 05873>	PROCEDURE	6C1V6
(float)	<nine, graphics, 05884>	PROCEDURE	6C1V7
(fmthou)	<nine, graphics, 01419>	EXT CONSTANT =5650140000000B	
(fmul)	<nine, graphics, 05859>	PROCEDURE	6C1V4
(fntc)	<nine, graphics, 01381>	EXT CONSTANT =0	6C1B3M
(fntd)	<nine, graphics, 01382>	EXT CONSTANT =1	6C1B3M
(fntf)	<nine, graphics, 01383>	EXT CONSTANT =2	6C1B3M
(fntn)	<nine, graphics, 01386>	EXT CONSTANT =5	6C1B3M
(fntnm)	<nine, graphics, 01385>	EXT CONSTANT =4	6C1B3M
(fnto)	<nine, graphics, 01384>	EXT CONSTANT =3	6C1B3M
(fntt)	<nine, graphics, 01387>	EXT CONSTANT =6	6C1B3M
(fone)	<nine, graphics, 01414>	EXT CONSTANT =2014000000000B	
(font)	<nine, graphics, 01377>	EXT CONSTANT =3	6C1B3L
(fpoint)	<nine, graphics, 01420>	EXT CONSTANT =1727053222568	
(fradd)	<nine, graphics, 05904>	PROCEDURE	6C1V9
(fsin)	<nine, graphics, 05911>	PROCEDURE	6C1V10
(fsqrt)	<nine, graphics, 05971>	PROCEDURE	6C1V13
(fsub)	<nine, graphics, 05852>	PROCEDURE	6C1V3
(ften)	<nine, graphics, 01417>	EXT CONSTANT =2045000000000B	
(fthou)	<nine, graphics, 01418>	EXT CONSTANT =2127640000000B	
(ftwo)	<nine, graphics, 01415>	EXT CONSTANT =2024000000000B	
(g4cdst)	<nine, graphics, 06026>	PROCEDURE	6C1W1
(galblk)	<nine, graphics, 04495>	PROCEDURE	6C1K1
(galist)	<nine, graphics, 04520>	PROCEDURE	6C1K5
(galoat)	<nine, graphics, 05469>	PROCEDURE	6C1S5
(galogi)	<nine, graphics, 04995>	PROCEDURE	6C1M15
(galowa)	<nine, graphics, 05264>	PROCEDURE	6C1Q4
(galstr)	<nine, graphics, 04508>	PROCEDURE	6C1K3
(gatbox)	<nine, graphics, 01300>	EXT	6C1B3A
(gatcd)	<nine, graphics, 01555>	FIELD - 36	6C1B13
(gatcl)	<nine, graphics, 01566>	EXT CONSTANT =1	6C1B13
(gatdsp)	<nine, graphics, 01557>	FIELD - 9	6C1B13
(gatexp)	<nine, graphics, 01563>	EXT CONSTANT =50	6C1B13
(gatfst)	<nine, graphics, 01569>	EXT CONSTANT =4	6C1B13
(gatgl)	<nine, graphics, 01567>	EXT CONSTANT =2	6C1B13
(gatml)	<nine, graphics, 01565>	EXT CONSTANT =0	6C1B13
(gatnew)	<nine, graphics, 01559>	FIELD - 1	6C1B13
(gatop)	<nine, graphics, 01558>	FIELD - 8	6C1B13
(gatptr)	<nine, graphics, 01568>	EXT CONSTANT =3	6C1B13
(gatsiz)	<nine, graphics, 01561>	EXT CONSTANT =100	6C1B13
(gatstid)	<nine, graphics, 01556>	FIELD - 36	6C1B13
(gattel)	<nine, graphics, 01564>	EXT CONSTANT =3	6C1B13
(gbadcoords)	<nine, graphics, 01218>	LOCAL	6B2A
(gbfsize)	<nine, graphics, 01483>	EXT CONSTANT =1200	6C1B9F
(gbiunt)	<nine, graphics, 01297>	EXT	6C1B3A
(gbldgi)	<nine, graphics, 04712>	PROCEDURE	6C1M7
(gbldtn)	<nine, graphics, 04808>	PROCEDURE	6C1M9
(gblrrc)	<nine, graphics, 01326>	EXT	6C1B3E
(gbottom)	<nine, graphics, 01636>	FIELD - 18	6C1B13
(gbstwa)	<nine, graphics, 05211>	PROCEDURE	6C1Q1
(gbuabb)	<nine, graphics, 01298>	EXT	6C1B3A
(gccdsx)	<nine, graphics, 06037>	PROCEDURE	6C1W2

(m22)	<nine, graphics, 01279>	EXT ADDRESS =4	6C1B2D
(m31)	<nine, graphics, 01280>	EXT ADDRESS =2	6C1B2E
(m32)	<nine, graphics, 01281>	EXT ADDRESS =5	6C1B2F
(maxfei)	<nine, graphics, 01478>	EXT CONSTANT =6	6C1B9A
(maxse)	<nine, graphics, 06017>	PROCEDURE	6C1V17
(minse)	<nine, graphics, 06021>	PROCEDURE	6C1V18
(noreast)	<nine, graphics, 01618>	EXT CONSTANT =1	
(north)	<nine, graphics, 01619>	EXT CONSTANT =2	
(norwest)	<nine, graphics, 01620>	EXT CONSTANT =3	
(point)	<nine, graphics, 01607>	EXT CONSTANT =5	
(rectangle)	<nine, graphics, 01605>	EXT CONSTANT =2	
(redge)	<nine, graphics, 01471>	EXT CONSTANT =84	6C1B7A
(scnbot)	<nine, graphics, 01444>	EXT	6C1B4C
(scnleft)	<nine, graphics, 01445>	EXT	6C1B4C
(scnright)	<nine, graphics, 01446>	EXT	6C1B4C
(scntop)	<nine, graphics, 01443>	EXT	6C1B4C
(scnvtop)	<nine, graphics, 01447>	EXT	6C1B4C
(sendbf)	<nine, graphics, 01482>	EXT	6C1B9E
(sendl)	<nine, graphics, 01345>	EXT	6C1B3H
(sendp)	<nine, graphics, 01344>	EXT	6C1B3H
(setmbottom)	<nine, graphics, 01303>	EXT	6C1B3B
(setmleft)	<nine, graphics, 01305>	EXT	6C1B3B
(setmright)	<nine, graphics, 01306>	EXT	6C1B3B
(setmtop)	<nine, graphics, 01302>	EXT	6C1B3B
(shcfw)	<nine, graphics, 06385>	PROCEDURE	6C1Z6
(singer)	<nine, graphics, 01336>	EXT	6C1B3F
(soueast)	<nine, graphics, 01624>	EXT CONSTANT =7	
(south)	<nine, graphics, 01623>	EXT CONSTANT =6	
(souwest)	<nine, graphics, 01622>	EXT CONSTANT =5	
(spie01)	<nine, graphics, 01363>	EXT	6C1B3J
(spie02)	<nine, graphics, 01364>	EXT	6C1B3J
(spie03)	<nine, graphics, 01365>	EXT	6C1B3J
(spie04)	<nine, graphics, 01366>	EXT	6C1B3J
(spie05)	<nine, graphics, 01367>	EXT	6C1B3J
(spie06)	<nine, graphics, 01368>	EXT	6C1B3J
(spie07)	<nine, graphics, 01369>	EXT	6C1B3J
(spieol)	<nine, graphics, 01370>	EXT	6C1B3J
(spifb)	<nine, graphics, 03754>	PROCEDURE	6C1G2
(spifel)	<nine, graphics, 03877>	PROCEDURE	6C1G6
(spijfn)	<nine, graphics, 01371>	EXT	6C1B3J
(spinc)	<nine, graphics, 03846>	PROCEDURE	6C1G5
(spird)	<nine, graphics, 03644>	PROCEDURE	6C1G1
(spirn)	<nine, graphics, 03774>	PROCEDURE	6C1G3
(spirs)	<nine, graphics, 03808>	PROCEDURE	6C1G4
(t4012)	<nine, graphics, 01333>	EXT	6C1B3F
(t4014)	<nine, graphics, 01334>	EXT	6C1B3F
(tedge)	<nine, graphics, 01472>	EXT CONSTANT =94	6C1B7A
(tekam)	<nine, graphics, 01424>	EXT CONSTANT =31	6C1B4A
(tekclr)	<nine, graphics, 01433>	EXT CONSTANT =12	6C1B4A
(tekcsize)	<nine, graphics, 01437>	EXT	6C1B4B
(tekesc)	<nine, graphics, 01431>	EXT CONSTANT =33B	6C1B4A
(tekgm)	<nine, graphics, 01423>	EXT CONSTANT =29	6C1B4A
(tekmode)	<nine, graphics, 01341>	EXT	6C1B3H
(tekpad)	<nine, graphics, 01429>	EXT CONSTANT =31B	6C1B4A
(tekrec)	<nine, graphics, 01701>	RECORD	6C1B13
(teksty)	<nine, graphics, 01432>	EXT CONSTANT =140B	6C1B4A

(tstgda)	<nine, graphics, 01488>	EXT	6C1B11
(typecalit)	<nine, graphics, 01509>	EXT CONSTANT =11	6C1B12
(typenulliit)	<nine, graphics, 01512>	EXT CONSTANT =14	6C1B12
(vcddash)	<nine, graphics, 01391>	EXT CONSTANT =3	6C1B3N
(vcdot)	<nine, graphics, 01390>	EXT CONSTANT =2	6C1B3N
(vcldash)	<nine, graphics, 01393>	EXT CONSTANT =5	6C1B3N
(vcom)	<nine, graphics, 01335>	EXT	6C1B3F
(vcsdash)	<nine, graphics, 01392>	EXT CONSTANT =4	6C1B3N
(vcsolid)	<nine, graphics, 01389>	EXT CONSTANT =1	6C1B3N
(west)	<nine, graphics, 01621>	EXT CONSTANT =4	
(wtarc)	<nine, graphics, 06255>	PROCEDURE	6C1X6
(wtchar)	<nine, graphics, 06242>	PROCEDURE	6C1X5
(wtline)	<nine, graphics, 06223>	PROCEDURE	6C1X3
(wtmark)	<nine, graphics, 06193>	PROCEDURE	6C1X1
(wtpnt)	<nine, graphics, 06234>	PROCEDURE	6C1X4
(wtrect)	<nine, graphics, 06209>	PROCEDURE	6C1X2
(xgannote)	<nine, graphics, 01713>	PROCEDURE	6C1D1
(xgchng)	<nine, graphics, 01742>	PROCEDURE	6C1D2
(xgcopy)	<nine, graphics, 01839>	PROCEDURE	6C1D3
(xgcout)	<nine, graphics, 01905>	PROCEDURE	6C1D4
(xgdraw)	<nine, graphics, 01943>	PROCEDURE	6C1D5
(xgerase)	<nine, graphics, 02099>	PROCEDURE	6C1D6
(xginit)	<nine, graphics, 02163>	PROCEDURE	6C1D7
(xginpt)	<nine, graphics, 02200>	PROCEDURE	6C1D8
(xglabl)	<nine, graphics, 02295>	PROCEDURE	6C1D9
(xglong)	<nine, graphics, 02335>	PROCEDURE	6C1D10
(xgmove)	<nine, graphics, 02374>	PROCEDURE	6C1D11
(xgorg)	<nine, graphics, 02440>	PROCEDURE	6C1D12
(xgport)	<nine, graphics, 02469>	PROCEDURE	6C1D13
(xgref)	<nine, graphics, 02506>	PROCEDURE	6C1D14
(xgrentr)	<nine, graphics, 02513>	PROCEDURE	6C1D15
(xgreset)	<nine, graphics, 02521>	PROCEDURE	6C1D16
(xgset)	<nine, graphics, 02536>	PROCEDURE	6C1D17
(xgshow)	<nine, graphics, 02693>	PROCEDURE	6C1D18
(xgterm)	<nine, graphics, 02751>	PROCEDURE	6C1D19
(xgtest)	<nine, graphics, 02774>	PROCEDURE	6C1D20
(xgtran)	<nine, graphics, 02793>	PROCEDURE	6C1D21
(xgupda)	<nine, graphics, 02800>	PROCEDURE	6C1D22
(xgvprt)	<nine, graphics, 02807>	PROCEDURE	6C1D23
(xgwdom)	<nine, graphics, 02897>	PROCEDURE	6C1D24

Graphics subsystem
  The Graphics subsystem enables you to create, display, modify, and print line drawings with text labels that are stored in NLS files. To use Graphics, you need a Graphics work station, consisting of a DNLS terminal and a Tektronix storage tube display. Every Graphics diagram is associated with an NLS statement and you can create them separately or in their appropriate place in a document. The diagrams can be printed on a photocomposition device or a special printer attached to the Graphics work station.
% Index in <nls,>
%FIRST SEARCHES
  Scratch
  This statement contains a scratch diagram (see attached diagram) 2A
  terminals
  ##<using>## 2B
  period (.)
  ##<show !grid>## 2C
  dot (.)
  ##<show !grid>## 2D
  Edges
  ##<Split>## 2E
  Margin
  ##<margins>## 2F
  Right
    A commandword that specifies the direction you wish a figure to face. See Insert Triangle, Insert Arrowhead. 2G
  Left
    A commandword that specifies the direction you wish a figure to face. See Insert Triangle, Insert Arrowhead. 2H
  Down
    A commandword that specifies the direction you wish a figure to face. See Insert Triangle, Insert Arrowhead. 2I
  Up
    A commandword that specifies the direction you wish a figure to face. See Insert Triangle, Insert Arrowhead. 2J
  Horizontally
    A commandword that specifies the way you want a circle to be drawn. See Insert Circle. 2K
  Vertically
    A commandword that specifies the way you want a circle to be drawn. See Insert Circle. 2L
How to use the Graphics subsystem 3
  Using the Textronix equipment with DNLS for Graphics 3A
    Textronix storage tube display
      To display diagrams with the Graphics subsystem, the Tektronix model 4014 or 4012 display must be connected to the Lineprocessor/alphanumeric terminal workstation via the Copy Printer port on a Lineprocessor specially configured for graphics use--a "Graphics Lineprocessor". The Tektronix's LINE/LOCAL switch must be set to LINE. The Tektronix display must warm up for several minutes before use. After you turn ON the power a bright green spot or pattern will appear on the Tektronix screen.
      Use the PAGE RESET key at the upper left hand corner of the Tektronix keyboard to clear the screen. Do this several times as the machine warms up. While the Tektronix display is turned on, it is in one of three modes: 3A1

CLEARED mode - no diagram display

Before you call for any diagram to be displayed there is nothing written on the Tektronix screen, although there may be spots of brightness. When you hit the PAGE RESET key, the screen remains in cleared mode (and the spots should dim). Once you display diagrams on the screen, hitting PAGE RESET will put the display back in cleared mode. 3A1A

VIEW mode - bright diagram display

When you create a new diagram or portray an existing one on the screen, a bright green image is written on the storage tube screen. When you are not actively working on the diagram, the image will remain bright for only 30 to 60 seconds. From VIEW mode, the display goes into HOLD mode. If you are interrupted during a command that manipulates the diagram, the screen will remain in VIEW mode indefinitely. This diminishes the rather limited life of the storage tube. Don't remain in VIEW mode for more than 15 minutes (do a Command Delete). 3A1B

HOLD mode - dim diagram display

The bright image on the screen will dim, though you can still perceive the lines, after a short while in VIEW mode. To get back in VIEW mode, hit the SHIFT key on the Tektronix keyboard, or move the cursor on the Tektronix screen. The HOLD mode consumes somewhat less of the tube-life than VIEW mode. Don't remain in HOLD mode for more than 60 minutes. Move the cursor back to the DNLS screen and press PAGE RESET to clear the screen before leaving the workstation. When you return, you'll have to portray the diagram again. 3A1C

Foot switch

Pressing the foot switch switches cursor movement back and forth between the alphanumeric and Tektronix screens. The foot switch is connected to the back of the Lineprocessor. You move the cursor on the alphanumeric screen in order to BUG textual entities. You can manipulate graphics entities on the Tektronix screen by bugging in the same way. To change the Tektronix from HOLD mode to CLEARED mode, move the cursor back to the alphanumeric display with the foot switch and hit PAGE RESET on the Tektronix. 3A2

Graphics output 3B

Hardcopy

Hardcopy pictures can be made of diagrams that are on the Textronix screen. When the Tektronix copier is attached to the Tektronix display terminal and powered ON, whatever is on the Tektronix screen (in VIEW mode) can be copied by pressing the copy button on the hard-copy unit or the Texktronix machine. The copier has to warm up for nearly 30 minutes before it makes uniform copies. If unsatisfactory copies appear after 30 minutes of power, adjust the contrast knob on the front of the copier. If the copies are still unsatisfactory, a person from Tektronix may need to make other adjustments. During copying the Tektronix display is not available to NLS. Wait until the paper cutter in the copier makes a firm click before entering any command which will write on the Tektronix. 3B1

Photocomposition

Diagrams, either by themselves or intermixed with the regular text of the NLS file, can be output to photocomposition

equipment, i.e., encoded on a magnetic tape which the photocomposer converts to microfilm or to photo-offset masters for printing. When you send a file with a diagram to a photocomposer, what is enclosed in the margins is the portion of the diagram that will be printed. The lines designating the margins on the screen will NOT be printed. 3B2

Viewing in Graphics 3C

Page space

"Page space" is the large storage area in which you "draw" the figures making up the diagram. You will probably view only a small portion of this area while working with a diagram. When you first create a diagram, a portion of the total page space will appear on the graphics screen enclosed in lines designating margins. The Tektronix screen is a "window" you can move around within your total page space. You can also vary the magnification of your view. Your view (and drawing) is not limited to the area within the margins, and each margin can be moved within page space. The Tektronix copier prints whatever is currently on the Tektronix screen, e.g., in the "window", including any visible margin lines. 3C1

Margins

Margins are characteristics of an individual diagram which stay with it until you change them. Margins for a new diagram enclose an 8.5 by 11 inch portion near the center of your page space. When you portray the diagram, a proportional representation of this area will nearly fill the window; two horizontal and two vertical lines represent the margins.

There will also be space outside the margins and you may draw in this area. You may expose area outside the margins by zooming to a different view of the page. Any drawings outside the current margins will not be output to COM, unless you move the margins outward to enclose them. The lines for the margins are not actually part of your diagram and won't appear in a photocomposed version. Turn the margins on and off with the Set Margin command. Change the location of the margins with the Set Margin (Right, Left, Top, Bottom) command. 3C2

Zooming

"Zooming" is changing the magnification of your view of your diagram. You may see part of the diagram in more detail--and thus can see less of the page space, or you may see more of the page space by showing less detail. In addition to changing the apparent sizes of figures in your window by zooming in or out, you may also use Zoom commands to focus on some other position in page space. These activities do not permanently affect the diagram. Use of the Set Margin and Zoom commands may expose sizable sections of page space outside the current margins, which are useful as a "scratch pad", to store templates for instance. 3C3

Split screens

Use edges in Graphics as you do in Base to set up additional windows on your Tektronix screen. Then each window may be used to view part of a diagram. You will be able to simultaneously view different diagrams or sections of a diagram in different magnifications. 3C4

Special terminology 3D

Drawspecs

"Drawspecs" are the elements that determine the characteristics of figures. This includes page size. The information listed below are the default Drawspecs for a newly created diagram. You can change drawspecs with a Set command. Page size will change when you change the margins with a Set Margin command. These are the default Drawspecs (in means inches): grid .1 in, resolution 1000 increments per inch, page size 8.5 in by 11 in [the number of inches between, the margins for the portrayed diagram], character height .166 in width .119 in (12 pt), left justification, TimesRoman font, Medium boldness, Straight, Not underlined, Proportional spacing, line type solid, line style regular, arrowhead height .111 in width .082 in 3D1

Figure

"Figure" is the general term that stands for all graphics entities. Figures may be referred to directly, such as circle, triangle, etc., or by the term Figure. Figure is used in the Insert, Copy, Move and Delete commands. When you see FIGURE in a command syntax, type one of the following as a commanword:

Arrowhead
Circle
Diamond
Figure
Line
Rectangle
Statement
Triangle 3D2

Statement

A graphics "Statement" refers to a string of text in the diagram itself that was inserted using the Insert Statement or Label comand You may specify font, empahis, and style for the characters and subsequently alter them with the Change Command. You insert a grahics statement within the bounds of a rectangle which shapes it. It caries that shape forever. The attaching point of a statement is the left edge of the upper leftmost character. 3D3

LINEWORK

"LINEWORK" stands for any graphics entity except a statement. When you see "LINEWO" in a command syntax, type one of the following as a commandword.

Arrowhead
Circle
Diamond
Figure
Line
Rectangle
Triangle 3D4

Attaching point

The "attaching point" of a figure is its leftmost visible point; if there is more than one leftmost point, the uppermost is the attaching point. Specify which figure you want (i.e. in a move or delete command) by bugging its attaching point. 3D5

Reference point

The "reference point" of a figure is used when moving or copying a figure or group of figures. You can choose any point on the figure as your reference point when a command prompts you. Then you specify where the reference point will lie in the new

location. 3D6

Group

A "Group" of figures is all those figures whose attaching points fall within a particular rectangular area. In the commands involving groups, you are asked to define the upper right and lower left corner of a rectangle that includes some attaching points. A group only remains defined for a single command; thus with each new command you can select a new group. 3D7

Grid

A "Grid" can be thought of as an imaginary piece of graph paper superimposing points on your diagram. These gridpoints are used to draw figures. Your cursor appears as a small lighted square which tracks around on the Tektronix screen as you move the mouse. When you bug, a small dot appears where the lower left corner of the cursor was caught at that moment, and the nearest gridpoint to that point will be taken as the location defined. You can have the grid show up as dots on sections of your screen. These dots are not actually part of your diagram and will disappear when your screen is refreshed. You can change the distance between the gridpoints with the Set Grid command. 3D8

BUG

"BUG" is an address in Graphics. When propmpted by B:, you should bug something on the Tektronix screen. Be sure that your foot switch has the cursor positioned on the Tektronix screen. The cursor is represented as a small rectangle. Use the lower left corner of the cursor as the point you bug with. 3D9

"see attached diagram"

The text "see attached diagram" is inserted as text in the NLS statement with which you have associated a diagram. This text is automatically inserted when you use the Create Diagram command. You may change this text using any of the commands in the Base subsystem.

Getting into graphics 3E

From TENEX

Call NLS 8.5, making sure that your terminal type is recognized as Lineprocessor so that you will be running DNLS:

@terminal type (is) LINEPROCESSOR <CR>
@nls <CR> 3E1

Loading the Graphics Subsystem

Graphics is a subsystem of NLS which you can load anytime with the Load Program command in the Programs subsystem. It is a very large program, so first delete any other programs you have loaded in your program buffer space:

BASE C: Goto (subsystem) Programs OK
PROGRAMS C: Delete All (programs in buffer) OK
Load Program GRAPHICS OK 3E2

Going to the Graphics Subsystem

After seeing "Subsystem GRAPHICS Now Available (Attached)" flash by in the TTY window, you may hereafter "Goto" Graphics anytime from any NLS subsystem:

Goto (subsystem) Graphics OK

Your herald is now GRAPHICS. The Graphics commands are now available to you along with the Universal commands such as Jump, Execute, Goto, Quit, Help, etc.

If you want to use some of the editing commands in Base to make

changes to the text of the file loaded, use the Goto Base command, do your editing, and then give the Quit command to return to Graphics. A portrayed diagram will remain on the screen in Hold mode (but be sure to foot-switch your cursor back to the alphanumeric screen). 3E3

Leaving the Graphics subsystem

To leave Graphics and erase your screen, use:

Quit To Base OK

Once you Quit from Graphics, you must hereafter Goto Graphics again (although during this NLS session you needn't Load Program again) in order to portray a diagram. 3E4

Commands 4

Attach (from the last line to) BUG OK

The Graphics command "Attach (from the last line to)" takes the end point of the last line you drew (even if you deleted it) and attaches a new line segment to that point. Parts of fiures such as rectangles or arrowheads do not count as new line segments. The new line is drawn horizontally or vertically depending on which distance is farthest from the original end point. 4A

What it's for

With this command you can either extend a line and know the extension lies along a straight path, or at right angles like building stairs. This attached line has its own attaching point and can be treated as a separate figure. Do not confuse this command with the Extend command which continues a single line in any direction, for example to draw a irregular polygon. The Extend command does NOT creat a new attaching point.

Backspace (the line at) BUG OK

The Graphics command "Backspace" allows you to delete the last line segment that you added to a line using the Extend command. BUG the attaching point of the line. You can use the Backspace command to delete all of the line segments you have added, but you may not backspace your original line. Use the Delete Line command for your original line. 4B

Change...

The Graphics command "Change" allows you to change your drawspecs. You can change the style or type of line used to draw a figure (except statements), the size of your characters, the font of your characters or the justification of a statement. Use the Show Specific command to verify changes that do not appear on your screen. 4C

Line: Change Line...

The Graphics command "Change Line" allows you to change either the style or the type of line in the LINEWORK you specify. 4C1

Style: Change Line Style...

You can change the style of the Graphics lines used to draw your diagrams according to one of the commands listed below. For LINEWORK specify anything that is a figure except statements; then BUG the LINEWORK you want changed. The LINEWORK with the new line style will be printed on top of the original and the latter will disappear when the screen is re-created. 4C1A

JChange Line Style (to) Very (bold) (for) LINEWORK (at) BUG OK

Change Line Style (to) Bold (for) LINEWORK (at) BUG OK

Change Line Style (to) Regular (for) LINEWORK (at) BUG OK

Change Line Style (to) Fine (for) LINEWORK (at) BUG OK
Change Line Style (to) Extra (fine) (for) LINEWORK (at) BUG OK

Type: Change Line Type...

You can change the line type used to draw your diagrams according to one of the Graphics commands listed below. For LINEWORK specify anything that is a figure except statements; then BUG the LINEWORK you want changed. The LINEWORK with the new line type will be printed on top of the original and the latter will disappear when the screen is re-created. 4C1B

]Change Line Type (to) Dot-dashed (for) LINEWORK (at) BUG OK
Change Line Type (to) Dotted (for) LINEWORK (at) BUG OK
Change Line Type (to) Long-dashed (for) LINEWORK (at) BUG OK
Change Line Type (to) Short-dashed (for) LINEWORK (at) BUG OK
Change Line Type (to) Solid (for) LINEWORK (at) BUG OK

Emphasis: Change Emphasis (to) TSTYLE (for statement at) BUG OK

The Graphics command "Change Emphasis" allows you to change type style of the characters in the statement you BUG. Specify one of the styles shown below. The statement with the new style will be printed or top of the original and the latter will disappear when the screen is re-created. 4C2

]Proportional (SPACING)
Monospaced
Not (UNDERLINED)
Straight
Slanted
Bold
Light
Medium

Font: Change Font (to) FONT (for statement at) BUG OK

The Graphics command "Change Font" allows you to change the type font of the characters in the statement you BUG. Specify one of the fonts shown below. The statement with the new font will be displayed directly over the original and the latter will only disappear when the screen is re-created. A different font will not actually appear as the Tektronix is not capable of displaying different fonts. Different fonts will appear when you produce the diagram using photocomposition. 4C3

]Timesroman
Newsgothic
Nmamicrofont
Ocrb
Film
Directory
Courier

Character: Change Character (height to) CONTENT (for statement at) BUG OK

The Graphics command "Change Character" allows you to change the height of the characters in the statement you BUG. You may give a number for height which will be taken as inches, a number followed by pt, which will be point size, or a number followed by a metric or English measure. You may not specify anything smaller that .001 inches. The statement with the new character

size will be displayed directly over the original and the latter will disappear when the screen is re-created. This command does not affect the drawspec setting for new statements, but changes the actual appearance of the statement that you bug. 4C4

Justification: Change Justification...

The Graphics command "Change Justification" allows you to change the justification of the particular statement that you BUG. Specify the justification you want according to one of the commands listed below. The newly justified statement will be printed on top of the original and the latter will disappear when the screen is re-created. 4C5

JChange Justification (to) Center (for statement at) BUG OK
Change Justification (to) Right (for statement at) BUG OK
Change Justification (to) Left (for statement at) BUG OK

Copy...

The Graphics command "Copy" allows you to copy a diagram, figure, or group of figures from one DESTINATION to another. 4D

Diagram: Copy Diagram (from statement) DESTINATION (to statement) DESTINATION OK: 4D1

Not yet implemented 4D1A

Group: Copy Group (of figures in the window at) BUG (and) BUG (from) BUG (to) BUG OK

The Graphics command "Copy Group" copies a group of figures that you specify from one place to another. You define the group by creating an imaginary rectangular window that encloses the attaching points of all the figures you want copied. First BUG the upper left corner of the rectangle and then BUG the lower right hand corner. After defining the group, BUG a reference point in the group followed by a BUG at a new point where you want the reference point to be when the figure is copied. 4D2

Figure: Copy FIGURE (at) BUG (from) BUG (to) BUG OK

The Graphics command "Copy FIGURE" allows you to copy any FIGURE from one place to another. First BUG the attaching point of the chosen figure. Next BUG a reference point on the figure and then BUG a new point where you want the reference point to be when the figure is copied. 4D3

%Figures

Arrowhead
Copy Arrowhead is a special case of Copy ##<copy !figure>## 4D3A1

Circle
Copy Circle is a special case of Copy ##<copy !figure>## 4D3A2

Diamond
Copy Diamond is a special case of Copy ##<copy !figure>## 4D3A3

Figure
Copy Figure is a special case of Copy ##<copy !figure>## 4D3A4

Line
Copy Line is a special case of Copy ##<copy !figure>## 4D3A5

Rectangle
Copy Rectangle is a special case of Copy ##<copy !figure>## 4D3A6

Statement
Copy Statement is a special case of Copy ##<copy !figure>##

In this case the text of the statement is copied including its layout. 4D3A7

Point
Copy Point is a special case of Copy ##<copy !figure>## 4D3A8

Triangle
Copy Triangle is a special case of Copy ##<copy !figure>## 4D3A9

Create Diagram (at statement) DESTINATION OK
The Graphics command "Create Diagram" associates a blank diagram with the statement in the NLS file you indicate for DESTINATION. The blank diagram (showing the margins only) will be portrayed in the window on your Tektronix screen. The text "(see attached diagram)" will automatically be written at the end of the statement you specify. This text is for your convenience in locating a diagram; you may edit it like other text in a statement. Graphics will "remember" that the diagram is attached in any case. 4E

Delete...
The Graphics commard "Delete" allows you to delete figures, groups and edges at the DESTINATIONS you specify. 4F

Diagram: Delete Diagram (at statement) DESTINATION OK
not implemented yet 4F1

Edge: Delete Edge (at) BUG OK
The Graphics command "Delete Edge" allows you to delete an edge between two windows and eliminates the contents of one of them from your screen. The contents of the window where your cursor is located during the final <CA> will fill the new larger window (the whole screen will be re-created). This may not be the same image you had before you split it originally. (Use Zoom Full to get the originally portrayed image.) 4F2

Group: Delete Group (of figures in the window at) BUG (and) BUG OK
The Graphics command "Delete Group" deletes the FIGURES in a group you define. You define a group by creating an imaginary rectangular window that encloses the attaching points of all the figures you want deleted. First BUG the upper left corner of the rectangle and then BUG the lower right hand corner. When you type the final <CA>, "X's" will be drawn on the attaching points of all figures that are deleted. The deleted figures will remain on the screen until it is re-created. 4F3

Figure: Delete FIGURE (at) BUG OK
The Graphics command "Delete FIGURE" deletes the FIGURE whose attaching point you BUG. An "X" will be drawn on the attaching point of the figure that is deleted. The deleted figure and the X will remain on the screen until it is re-created. 4F4

%Figures

Arrowhead:
Delete Arrowhead is a special case of Delete ##<delete !figure>## 4F4A1

Circle:
Delete Circle is a special case of Delete ##<delete !figure># 4F4A2

Diamond:
Delete Diamond is a special case of Delete ##<delete !figure>## 4F4A3

Line:

Delete Line is a special case of Delete ##<delete !figure>## 4F4A4
Point:
Delete Point is a special case of Delete ##<delete !figure>## 4F4A5
Rectangle:
Delete Rectangle is a special case of Delete ##<delete !figure>## 4F4A6
Rectangle;
Delete Statement is a special case of Delete ##<delete !figure>## 4F4A7
Triangle:
Delete Triangle is a special case of Delete ##<delete !figure>## 4F4A8

Handle OK 4G

Input (SPL diagram from file) DESTINATION (Blocknumber) BLKNUM (at statement)DESTINATION

If you have prepared an SPL file, this command allows you to move a diagram from that file into NLS Graphics. The first DESTINATION requires the name of a seguntial file; BLKNUM requires a formating number you have set in the SPL file; the second DESTINATION is the NLS statement where the diagram will reside. Conversion of graphics from other systems is not difficult for some one with a moderate knowledge of relevant programming. For more information see SPL. 4H

Insert...

The Graphics command "Insert" allows you to create or draw figures or add text to diagrams that are associated with statements in an NLS file. 4I

BUG: Insert BUG (and) BUG OK

The Graphics command "Insert" draws for you either a horizontal line or a vertical line depending on the points that you bug. The line begins at the POINT you first BUG. The system draws imaginary horizontal and vertical lines through the second POINT you BUG. Your first POINT will be connected to either the horizontal or vertical line depending on which distance is greater. 4I1

Edge: Insert Edge (perpendicular to)...

The Graphics commands "Insert Edge" splits your screen into separate windows according to one of the commands listed below. The division is indicated by a double line. The entire contents of the current window will be proportionally squeezed into the side of the division in which your cursor is located at the final <CA>. (Depending on proportions, you may even see more of page space). The other window will be empty; it can be filled using the Portray command. Manipulate the contents of any new windows in the same manner you've been using. Commands will affect the contents of the window in which your cursor is located when you BUG, or give the final CA. Since Each window has four edges, it can be further split either horizontally or vertically. You can move or copy figures from window to window by bugging in those locations. 4I2

Insert Edge (perpendicular to) BUG OK

The Graphics command "Insert Edge" works like the Base command "Insert Edge."##<base, insert !edge>## 4I2A

Insert Edge (perpendicular to) Center (of) BUG OK

The Graphics command "Insert Edge" is not implemented at

this time. 4I2B

Statement: Insert Statement (into the window at) BUG (and) BUG CONTENT OK

The Graphics command "Insert Statement" allows you to insert text into your diagram. Indicate the rectangle you want your text enclosed in by BUGGING the upper left corner and then the lower right corner. Then, type in the text. As much of the text as possible will be fit in the rectangle with lines being wrapped around where necessary. If, however, the rectangle cannot contain enough lines, the text will be truncated. This rectangle of text will then be referred to as a statement or Figure; its attaching point is the left edge of the upper leftmost character. You may also use the Label command to insert text. 4I3

Circle: Insert Circle (center) BUG BUG OK

The Graphics command "Insert Circle" draws a circle with the center at the first point you BUG and which passes through the second point you BUG. 4I4

Horizontally: Insert Circle (center) BUG Horizontally (tangent to) BUG OK

After specifying the center, type "h" for horizontally. BUG a point through which an imaginary horizontal line will be drawn. Your circle will be drawn tangent to (resting against) that horizontal line. 4I4A

Vertically: Insert Circle (center) BUG Vertically (tangent to) BUG OK

After specifying the center, type "v" for vertically. BUG a point through which an imaginary vertical line will be drawn. Your circle will be drawn tangent to (resting against) that vertical line. 4I4B

Triangle: Insert Triangle (pointing)...

The group of Graphics commands "Insert Triangle (pointing)..." draws a triangle with the top pointing in the direction you specify according to one of the commands below. First BUG the vertex (called top in the command) of the traingle and then BUG one base point of the traingle. An isosceles triangle will then be drawn. 4I5

JRight: Insert Triangle (pointing) Right (top) BUG (one base point at) BUG OK

Left: Insert Triangle (pointing) Left (top) BUG (one base point at) BUG OK

Down: Insert Triangle (pointing) Down (top) BUG (one base point at) OK BUG

Up: Insert Triangle (pointing) Up (top) BUG (one base point at) BUG OK

Arrowhead: Insert Arrowhead (pointing)...

The Graphics command "Insert Arrowhead" draws an arrowhead in the direction you specify by using one of the commands below. BUG the point where you want the tip of your arrowhead. You may spcify your arrowhead to be different sizes. See Set Arrowhead Size. 4I6

JRight: Insert Arrowhead (pointing) Right (at) BUG OK

Left: Insert Arrowhead (pointing) Left (at) BUG OK

Down: Insert Arrowhead (pointing) Down (at) BUG OK

Up: Insert Arrowhead (pointing) Up (at) BUG OK

Diamond: Insert Diamond (top at) BUG (bottom at) BUG (side at) BUG OK

The Graphics command "Insert Diamond" draws a diamond of the size you specify and at the loction you designate. First BUG where you want the top point of the diamond to be. Then BUG a point through which an imaginary horizontal line is drawn. The bottom of the diamond will pass through this line. Finally, BUG a point through which an imaginary vertical line is drawn. One side point of the diamond will pass through this line. 4I7

Square: Insert Square (upper left corner at) BUG (size) BUG OK

The Graphics command "Insert Square" draws a square of the size you specify at the location you designate. First BUG the point you want for the upper left corner. Approximate one of the other three corners with the second BUG. The system enables this approximation by determining imaginary horizontal and vertical lines through the second point bugged. One side of the square will be drawn, horizontally or vertically, from the upper left point to a point on either of those lines, whichever is farther away. The other equal sides of the square are determined from this first side. When you wish to Copy, Move, or Delete a Square, you must specify either Rectangle or Figure as the word Square will not be recognized. 4I8

Rectangle: Insert Rectangle (corner at) BUG (opposite at) BUG OK

The Graphics command "Insert Rectangle" allows you to draw a rectangle of the size you specify and at the location you designate. First BUG the point you want for one of the corners, and then BUG the point you want for the corner that is diagonally opposite to your first BUG. 4I9

Vertical: Insert Vertical (line starting at) BUG (ending) BUG OK

The Graphics command "Insert Vertical" allows you to draw a vertical line. The line begins at the first point you BUG. An imaginary horizontal line is drawn through the second point you BUG. The system connects your first point to that line by drawing a vertical line. Your second BUG approximates the vertical endpoint. 4I10

Horizontal: Insert Horizontal (line starting at) BUG (ending) BUG OK

The Graphics command "Insert Horizontal" allows you to draw a horizontal line. The line begins at the first point you BUG. An imaginary vertical line is drawn through the second point you BUG. The system connects your first point to that line by drawing a horizontal line. Your second bug approximates the horizontal endpoint. 4I11

Line: Insert Line (between) BUG (and) BUG OK

The Graphics command "Insert Line" allows you to draw a line between any two points that you BUG. The line drawn is not necessarily horizontal or vertical. 4I12

Point: Insert Point (at) BUG OK

The Graphics command "Insert Point" allows you to insert a POINT at the place you BUG. The nearest grid point will be taken. 4I13

Label (diagram at) BUG (with) CONTENT

The Graphics command "Label" associates some text with the diagram you BUG. The text you give when prompted for CONTENT will be written on the graphics screen. It will be justified according to the justification that you have set. Text inserted with the Label command will not be truncated or wrapped around unless you include a

<CR>. The inserted text is subsequently referred to as a statement or a Figure; its attaching point is the upper leftmost character.
see statement 4J
Move...
The Graphics command "Move" allows you to move an edge, a figure, or a group of figures from one place to another. 4K
Diagram: Move Diagram (from statement) DESTINATION (to statement) DESTINATION OK
The Graphics command "Move Diagram" is not yet implemented. 4K1
Edge: Move Edge (from) BUG (to) BUG OK:
The Graphics command "Move Edge" works the same as the Base command "Move Edge." ##<base, move !edge>## 4K2
Group: Move Group (of figures in the window at) BUG (and) BUG (from) BUG (to) BUG OK
The Graphics command "Move Group" moves a group of figures that you specify from one place to another. You define the group by creating an imaginary rectangular window which encloses the attaching points of all the figures you want moved. First BUG the upper left corner of the rectangle and then BUG the lower right hand corner. After defining the group, BUG a reference point in the group followed by a BUG where you want the new reference point to be when the group is moved. When the group is moved, it will appear in the new location and X's will be drawn on the attaching points of the figures in the old location. The figures in the old location and the X's will remain until the screen is re-created. 4K3
Figure: Move FIGURE (at) BUG (from) BUG (to) BUG OK
The Graphics command "Move FIGURE" allows you to move any FIGURE from one place to another. First BUG the attaching point of the chosen figure. Next BUG a reference point, and then BUG a point where you want the new reference point to be when the figure is moved. The figure will appear in the new location and an "X" will be drawn on the attaching point of the figure in the old location. The old figure and the "X" will remain until the screen is re-created. 4K4
%Figures
Arrowhead:
Move Arrowhead is a special case of Move ##<move !figure>## 4K4A1
Circle:
Move Circle is a special case of Move ##<move !figure>## 4K4A2
Diamond:
Move Diamond is a special case of Move ##<move !figure>## 4K4A3
Line:
Move Line is a special case of Move ##<move !figure># 4K4A4
Rectangle:
Move Rectangle is a special case of Move ##<move !figure># 4K4A5
Point:
Move Point is a special case of Move ##<move !figure> 4K4A6
Triangle:
Move Triangle is a special case of Move ##<move !figure># 4K4A7
Output (diagram at) DESTINATION (to COM file) CONTENT

The Graphics command "Output" takes the diagram you specify for DESTINATION and places it in the COM file that you specify for CONTENT. This enables you to transmit an individual diagram to COM. 4L

Portray (diagram at statement) DESTINATION (in graphics window) BUG

The Graphics command "Portray"" allows you to see the diagram that is associated with a statement in an NLS file at the DESTINATION you specify. It will be shown in the graphics window you BUG. 4N

Reset Drawspecs OK

The Graphics command "Reset Drawspecs" changes all drawspecs for figures to be inserted--grid size, arrowhead size, character size, font, character emphasis, justification, and line style and type--back to the initial settings for a Graphics session. See the Show Drawspecs command for the default settings. Reset Drawspecs does not affect page size or the figures already in the diagram. 4N

Set...

The Graphics command "Set" allows you to specify different characteristics for your drawspecs. They will remain inforce as you work and from session to session until you reset them. Use the Show Drawspecs command to learn what drawspecs are inforce. To change the charactertics of individual figures, use the Change command. The Show Specif command lists the characteritcs of individual figures. 4O

Character: Set Character (height) CONTENT

The Graphics command "Set Character (height)" allows you to specify the size of the letters you subsequently insert into your diagrams as statements and labels. The default size is 12 point type (shown in drawspecs as height .166 in width .199 in). You may specify a number which will be taken as inches, a number followed by pt, which will be point size, or a number followed by a metric or English measure. You may not specify anything smaller that .001 inches. For a particular height, the proper corresponding width will be set; you cannot set the width (though it will show with the Show Drawspecs command). 401

Emphasis: Set Emphasis TSTYLE OK

The Graphics command "Set Emphasis" allows you to set the type style of the characters in the statements to be subsequently inserted into you diagrams during this Graphics session. Specify one of the styles shown below. The default emphasis is medium boldness, proportional spacing, and not underlined. 402

]Proportional (SPACING)
Monospaced
Not (UNDERLINED)
Straight
Slanted
Bold
Light
Medium

Font: Set Font FONT OK

The Graphics command "Set Font" allows you to specify the type face of the statements to be subsequently inserted into your diagrams during this Graphics session (with the Label or Insert Statement command). The initial setting is Times Roman font. The Tektronix can display only one font, so you will not be able to examine the results of using different fonts until the diagram is output to COM. However, the Show Specific command will display the name of the font for a particular statement that's

already part of the diagram. Specify one of the fonts shown below: 403

]Timesroman
Newsgothic
Nmamicrofont
Ocrb
Film
Directory
Courier

Arrowhead: Set Arrowhead (height) CONTENT (width) CONTENT

The Graphics command "Set Arrowhead" allows you to specify the size of the arrowhead. The default size is a height of 8 point and a width of 6 point (shown in drawspecs as height .111 in width .082 in). You may set your arrowhead size in terms of point size, English or metric measures. 404

Grid: Set Grid (size) CONTENT

The Graphics command "Set Grid" allows you to specify the size of the grid. The default grid size is .1 inches which means there is a grid point every tenth of an inch. You may set your grid size in terms of point size, English or metric measures. You may not set your grid size to smaller than .001 inches. Use the "Show Grid" command to see a portion of the grid you are using. 405

Justification: Set Justification

The Graphics command "Set Justification" allows you to define how you want the text that is subsequently inserted to appear in the diagram. You can justify the text according to one of the commands below. The default justification is left. Justification effects the text that you insert with either the Insert Statement or Label command. 406

]Set Justification Center OK
Set Justification Right OK
Set Justification Left OK

Margin: Set Margin...

The Graphics command "Set Margin" allows you to define the position of each margin and/or to display or not to display the margin according to one of the commands listed below. When using the commands to define the position of a margin, BUG a point through which the new margin will pass. The new margin will be shown, but the old one will also remain until the screen is re-created. The diagram then has the new margins. The default margins are an eight and one half by eleven inch page. When you Set Margin Off, your margins will no longer be displayed. This is most useful when you wish to use the hardcopy machine and do not want margins to show. When you Set Margin On, the margins will not reappear until you recreate the screen. 407

]Set Margin Right BUG OK
Set Margin Left BUG OK
Set Margin Top BUG OK
Set Margin Bottom BUG OK
Set Margin On OK
Set Margin Off OK
Resetting margins
It is necessary to reset margins by bugging the correct location. It is often useful to zoom out and use the Show Grid command to get a view where it is handy to set margins. 407B

Line...

The Graphics commands "Set Line" allow you to set either the style or the type of line you want when drawing your LINEWORK. 408

Style: Set Line Style...

You can set the style of the lines used to draw your diagrams according to one of the Graphics commands listed below. The default line style is regular. 408A

]Set Line Style (to) Very (bold) OK
Set Line Style (to) Bold OK
Set Line Style (to) Regular OK
Set Line Style (to) Fine OK
Set Line Style (to) Extra (fine) OK

Type: Set Line Type...

You can set the type of line used to draw your diagrams according to one of the Graphics commands listed below. The default line type is solid. 408B

]Set Line Type (to) Dot-dashed OK
Set Line Type (to) Dotted OK
Set Line Type (to) Long-dashed OK
Set Line Type (to) Short-dashed OK
Set Line Type (to) Solid OK

Terminal: Set Terminal...

The Graphics commands "Set Terminal" allow you to specify the type of terminal you have according to one of the following commands. If you have a Textronix 4012, set your terminal type to T2. If you have a Textronix 4014, set your terminal type to T4 (which is also the default.) 409

]Set Terminal (type) T2 OK
Set Terminal (type) T4 OK

Viewgraph: Set Viewgraph...

The Graphics commands "Set Viewgraph" allow you to have all the linework and text be drawn thicker and darker on your screen. Specify one of the commands below. When Viewgraph mode is ON, and the screen is refreshed, all linework and text will be drawn four times. This is useful when a viewgraph is to be made directly from the Tektronix copier's output. Characters of statements that are quite small or very close together on the screen may be so thickly drawn that they will be indistinct. Viewgraph mode is Off by default. 4010

]Set Viewgraph (mode on/off) Off OK
Set Viewgraph (mode on/off) On OK

Show...

The Graphics command "Show" allows you to display different kinds of information about the figures in your diagrams. 4P

Drawspecs: Show Drawspecs OK

The Graphics command "Show Drawspecs" shows the Drawspecs that are currently determining the characteristics of figures to be inserted. The page size for the currently portrayed diagram is also included. The default Drawspecs for a newly created diagram are listed below. After you use a Set command to change Drawspec settings, those changes will be reflected in this command. Page size will change when you change the margins for the diagram portrayed (with Set Margin commands). These are the default Drawspecs ("in" means inches): grid .1 in, resolution 1000 increments per inch, page size 8.5 in by 11 in [the number of inches between, the margins for the portrayed diagram], character

height .166 in width .119 in (12 pt), left justification, TimesRoman font, Medium boldness, Straight, Not underlined, Proportional spacing, line type solid, line style regular, arrowhead height .111 in width .082 in 4P1

Grid: Show Grid (in the window) BUG (other corner) BUG OK
or
.(Show Grid in the window) BUG (other corner) BUG OK

The Graphics command "Show Grid" or typing a period (.) displays a grid on your screen. First BUG the upper left corner of the rectangle where you want to display the grid and then BUG the lower right corner of this rectangle. All grid points will be shown. You can type a <CTRL-O> when you want the grid to stop being shown. These points are not part of your diagram and will not continue to appear on the screen when it is re-created. The grid points show you the exact points that will be taken when you BUG a spot. 4P2

Specific: Show Specific (drawspecs for) FIGURE (at) BUG OK

The Graphics command "Show Specific" displays the Drawspecs that were used to draw any particular FIGURE already in the diagram. BUG the attaching point of the FIGURE. Do not confuse this command with the Show Drawspecs command. If you want to change any of the drawspecs of the FIGURE, use a Change command. 4P3

%Figures

Arrowhead
Show Specific (drawspecs for) Arrowhead is a special case of Show Specific ##<show !specific>## 4P3A1

Circle
Show Specific (drawspecs for) Circle is a special case of Show Specific ##<show !specific>## 4P3A2

Diamond
Show Specific (drawspecs for) Diamond is a special case of Show Specific ##<show !specific>## 4P3A3

Figure
Show Specific (drawspecs for) Figure is a special case of Show Specific ##<show !specific>## 4P3A4

Line
Show Specific (drawspecs for) Line is a special case of Show Specific ##<show !specific>## 4P3A5

Rectangle
Show Specific (drawspecs for) Rectangle is a special case of Show Specific ##<show !specific>## 4P3A6

Point
Show Specific (drawspecs for) Point is a special case of Show Specific ##<show !specific>## 4P3A7

Triangle
Show Specific (drawspecs for) Triangle is a special case of Show Specific ##<show !specific>## 4P3A8

Test (*** Configuration Checkout ***) OK

The Graphics command "Test" displays on your Tektronix screen a test pattern. You do not have to have a diagram portrayed. If this command does not work you probably have some some hardware problem. 4Q

Universal commands available in all subsystems
##<nls, universal>## 4R
Update (graphics display) OK

The Graphics command "Update" rewrites the screen with all the changes you have made to your diagram since the last update. This command is helpful after you have used the Move, Delete, Change, and Show Grid commands. The Graphics Update command re-creates the display so you can clearly see your changes to the diagram. Changes to a diagram are treated as modifications to the file just like other edits of NLS files. The Graphics Update command DOES NOT merge the modifications into another version of the NLS file; execute the Update File command in BASE to update the file, i.e., to make the re-created changes permanently part of the file. 4S

Xtend...

The Graphics command "Xtend" allows you to continue a line in one of two ways. 4T

Xtend Visibly (the line at) BUG (to) BUG OK

The Graphics command "Xtend Visibly" allows you to continue the line whose attaching point you BUG, to the second point you BUG. The original line and any extended pieces are a single entity with one attaching point. Note that your attaching point changes if you extend the line further left of or exactly above your last attaching point. The command Backspace deletes the last extension to the line. You cannot use Backspace to delete the original line. Do not confuse this with the Attach command. 4T1

Xtend Invisibly (the line at) BUG (to) BUG OK

The Graphics command "Xtend Invisibly" allows you to continue the line whose attaching point you BUG, to the second point you BUG. The new segment you specify will not be visible. The original line and any extended pieces are a single entity with one attaching point. Note that your attaching point DOES NOT change if you invisibly extend the line further left of or exactly above your last attaching point. The command Backspace deletes the last extension to the line. You cannot use Backspace to delete the original line. Do not confuse this with the Attach command. 4T2

Zoom...

The Graphics command "Zoom" allows you to view your diagrams in different ways. 4U

In: Zoom (or move image) In OK

The Graphics command "Zoom (or move image) In" allows you to get a more detailed look at part of your diagram. All figures that are located around the center point on your graphics display are blown up to twice their current size. You will not be able to see all of the figures that were previously displayed. 4U1

To: Zoom (or move image) To (window) (one corner) BUG (other corner) OK BUG

The Graphics command "Zoom (or move image) To" allows you to get a more detailed view of the part of your diagram that you specify. First BUG one corner of an imaginary rectangle that will enclose the part of the diagram that you want to view in more detail. Then BUG the lower right corner of this rectangle. When you type <CA>, the rectangle you have defined will be enlarged to fill your entire screen. 4U2

Full: Zoom (or move image) Full (size of the margins) OK

The Graphics command "Zoom (or move image) Full" fills your screen so that your margins are as near as porportionally possible to the edges of the screen. Therefore, if you move your margins and Zoom to Full, a different portion of the diagram will

be seen on your screen. This is the same view as you would get in response to the Portray command; and this command also has the same effect as the Update command. Zoom to Full is useful after having used some other Zoom commands. 4U3

Out: Zoom (or move image) Out OK

The Graphics command "Zoom (or move image) Out" allows you to get a more global view of your diagram. All of the figures on your screen are reduced in size by half and thus your screen encloses more of the entire diagram. The center point of your window remains the same. 4U4

Center: Zoom (or move image) BUG (to) Center OK

The Graphics Command "Zoom (or move image) BUG (to) Center allows you to specify with a BUG which point on the diagram you wish to be moved to the center of your window. This changes the position of your view, not the magnification. 4U5

BUG: Zoom (or move image) BUG (to) BUG OK

The Graphics command "Zoom (or move image)" allows you to move around the part of your diagram that is on your screen. First BUG any point on your screen. Next, BUG the point where you want your first BUG to be located after the screen is re-created. The size of all the figures will remain the same; their position on the screen may change. 4U6

%LAST SEARCHES

spl

Sequential Plot Language (SPL) provides a means of converting graphical material created on systems other than NLS into NLS diagrams, which can then be modified and printed in the same way as diagrams created through the graphics editor within NLS. 5A

Syntax of Commands

SPL files consist of a list of command characters, coordinates, and text strings. The files are free format, that is, a wide variation in punctuation and organization is permitted. Commands begin with a character, which is followed by an ordered list of numbers and strings, and are terminated by semi-colons. Punctuation between the numbers and strings may be any character except a number.

Numbers consist of the digits 0-9 and the plus and minus signs. The value is interpreted as a decimal integer in page coordinates. The fractional part (that is, any digits following a period within the number) is ignored. The user may select page coordinates in the range + or - 128,000. For example, the user may treat the page system as thousands of an inch. In this case, the value 8500.98 will mean 8.5 inches (not 8.50098 inches). If the page system is visualized as feet, then the number 5280 will mean one mile.

Strings consist of any text punctuated by single quote marks. Any text may be included. If a single quote is required it may be entered by immediately following it with another single quote (for example 'Don''t').

Command Set

Block Command

Groups of commands may be organized into blocks and numbered. This allows the user to put several diagrams on a single sequential plot file. The Graphics command "Input (SPL file)" requests a block number which is used to access the SPL block delimited by a block command. The command has the form:

```
      # block-number ;
      Margin Command
         The "margin" command allows the user to set the coordinates
      in the page system which bound the region of interest.  (This
      command sets the NLS Graphics margins.)It has the format:
          M left right top bottom;
         The command character may be upper- or lower-case and the
      command may be partially or fully spelled out.  For example,
      M, m, marg, margin, and MARGIN are identical as far as SPL is
      concerned.
      Line Command
         The "line" command allows the user to input broken lines of
      up to 250 segments.  The first coordinate is treated as the
      base of the line. It has the format:
           L x1 y1 x2 y2 ... x250 y250;
      A minimum of two points is required.
      Text Command
      The "text" command allows for the insertion of text at a
      specific position and height in the page coordinate system.
      It has the format:
      T x y height 'string';
      String may contain up to 72 characters.
   Errors
      We assume that the sequential file will be written by higher
   level user routines in the user's own language and system.  As a
   result, the error reporting is by class only.  Error messages are
   appended to the destination statement by the SPL system when the
   "Input (SPL file)" command is used.
   Language Interfaces
       At present, interfaces are available for both BASIC and
   FORTRAN.  The FORTRAN system is compatible with the CALCOMP plot
   package, which is widely used for most static plotting.
%source-code%                                                        6
   %grammar%                                                        6A
      FILE graphicsgrammar % (arcsubsys,cgcml,) (relnine,
      graphics.cml,) %
      % COMPILE INSTRUCTIONS %                                     6A2
         INCLUDE <nine, nls-grammar, flags !subsystems>           6A2A
      % DECLARATIONS %                                            6A3
         INCLUDE <nine, nls-grammar, declarations !universal>     6A3A
         DECLARE FUNCTION                                         6A3B
            xgannote, xgchng, xgcopy, xgcout, xgdraw, xgerase, xginit,
            xginpt, xglabl, xglong, xgmove, xgorg, xgport, xgset,
            xgshow, xgtest, xgupda, xgvprt, xgwdom, xgrentr, xgreset,
            xgterm;                                              6A3B1
         DECLARE COMMAND WORD
            % selectors %
               "BUG" = 33 SELECTOR
                  POINT = gpgetcoords,                          6A3C1A1
            "CELL" = 70,
            "SUBCELL" = 71,
            "COLLECTION" = 72,
            "DIAGRAM" = 79,
            "LEFT" = 83,
            "RIGHT" = 84,
            "CENTER" = 85,
```

```
"ALONG" = 86,
"HORIZONTALLY" = 87,
"VERTICALLY" = 88,
"TOUCHING" = 89,
"IN" = 90,
"OUT" = 91,
"TO" = 92,
"FULL" = 93,
"TOP" = 94,
"BOTTOM" = 95,
"VIEWGRAPH" = 122,
"PERPENDICULAR" = 96,
"FIGURE" = 100,
"RECTANGLE" = 101,
"LINE" = 102,
"UNCONSTRAINED" = 103,
"POINT" = 104,
"CIRCLE" = 105,
"DIAMOND" = 106,
"ITRIANGLE" = 107,
"TRIANGLE" = 107,
"ARROWHEAD" = 108,
"ANNOTATION" = 109,
"SQUARE" = 110,
"JUSTIFICATION" = 10,
"ASSOCIATE" = 11,
"DISSOCIATE" = 12,
"PARTITION" = 13,
"MOVE" = 14,
"ERASE" = 15,
"MARGIN" = 16,
"GRID" = 17,
"DRAWSPECS" = 18,
"CONTINUATION" = 19,
"EXTEND" = 20,
"VISIBLY" = 20,
"INVISIBLY" = 22,
"BACKSPACE" = 21,
"RESOLUTION" = 22,
"BASIC" = 23,
"UNIT" = 24,
"FONT" = 25,
"TYPE" = 26,
"STYLE" = 27,
"TERMINAL" = 28,
"SPECIFIC" = 30,
"EXTRA" = 1,
"FINE" = 2,
"REGULAR" = 3,
"BOLD" = 4,
"VERY" = 5,
"SOLID" = 1,
"DOTTED" = 2,
"DOT-DASHED" =3,
"SHORT-DASHED" = 4,
"LONG-DASHED" = 5,
```

```
        "COURIER" = 1,
        "DIRECTORY" = 2,
        "FILM" = 3,
        "OCRB" = 4,
        "NMAMICROFONT" = 5,
        "NEWSGOTHIC" = 6,
        "TIMESROMAN" = 7,
        "T4" = 1,
        "T2" = 2,
        "EMPHASIS" = 29,
        "MEDIUM" = 1,
        "LIGHT" = 3,
        %"BOLD" = 4,%
        "SLANTED" = 5,
        "STRAIGHT" = 6,
        "UNDERLINED" = 7,
        "NOT" = 8,
        "MONOSPACED" = 9,
        "PROPORTIONAL" = 10,
        "ONE" = 1,
        "TWO" = 2,
        "THREE" = 3;
    DECLARE VARIABLE
        s1, s2, namfil;
% COMMON RULES %
    INCLUDE <nine, nls-grammar, rules !universal>
    bigdirection = "UP" / "DOWN" / "LEFT" / "RIGHT";
    margin = "TOP" / "BOTTOM" / "LEFT" / "RIGHT";
    tstyle = "MEDIUM" / "LIGHT" / "BOLD" / "SLANTED" / "STRAIGHT"
    !L2! / "UNDERLINED" / "NOT" <"UNDERLINED"> / "MONOSPACED" !L2!
    / "PROPORTIONAL" <"SPACING">;
    ltype = "SOLID" / "DOTTED" / "DOT-DASHED" !L2! /
    "SHORT-DASHED" !L2! / "LONG-DASHED";
    lstyle = "EXTRA" <"fine"> / "FINE" / "REGULAR" / "BOLD" /
    "VERY" <"bold">;
    font = "COURIER" / "DIRECTORY" / "FILM" / "OCRB"/
    "NMAMICROFONT" !L2! / "NEWSGOTHIC" / "TIMESROMAN";
    justify = "LEFT" / "RIGHT" / "CENTER";
    grotate = "ONE" / "TWO" / "THREE" !L2!;
    figure = "ARROWHEAD" / "CIRCLE" / "DIAMOND" / "FIGURE" /
    "TRIANGLE" / "LINE" / "POINT" / "RECTANGLE" / "STATEMENT";
    linework = "ARROWHEAD" / "CIRCLE" / "DIAMOND" / "FIGURE" /
    "TRIANGLE" / "LINE" / "POINT" / "RECTANGLE";
% COMMANDS % SUBSYSTEM graphics KEYWORD "GRAPHICS"
    INITIALIZATION
        zinit = xginit();
    RENTRY
        zrentr = xgrentr();
    TERMINATION
        zterm = xgterm();
    COMMAND zattachd =
        "ATTACH"
        <"from the last line to">
        s1 _ #"CONTINUATION"
        s2 _ NULL
        param _ DSEL ( #"BUG" )
```

```
      param2 _ NULL
      param3 _ NULL
      CONFIRM
      xgdraw(s1,s2,param,param2,param3)
      ;
   COMMAND zback =
      "BACKSPACE"
         s1 _ #"BACKSPACE"
         <"the line at">
         param _ DSEL ( #"BUG" )
         param2 _ NULL
      CONFIRM
      xglong(s1,param,param2)
      ;
   COMMAND zchange =
      "CHANGE" !L2!
         (s1 _ "JUSTIFICATION" <"to">
            param _ justify <"for statement at">
            param2 _ DSEL ( #"BUG" )
            s2 _ NULL
         /s1 _ "CHARACTER" <"height to">
            param _ LSEL(#"TEXT") <"for statement at">
            param2 _ DSEL ( #"BUG" )
            s2 _ NULL
         /s1 _ "FONT" <"to">
            param _ font <"for statement at">
            param2 _ DSEL ( #"BUG" )
            s2 _ NULL
         /s1 _ "EMPHASIS" <"to">
            param _ tstyle <"for statement at">
            param2 _ DSEL ( #"BUG" )
            s2 _ NULL
         / "LINE"
            (s1 _ "TYPE" <"to"> param _ ltype
            /s1 _ "STYLE" <"to"> param _ lstyle)
            <"for"> s2 _ linework <"at">
            param2 _ DSEL ( #"BUG" )
         )
      CONFIRM
      xgchng(s1,s2,param,param2)
      ;
   COMMAND zcopyd =
      "COPY"
         param3 _ NULL
         s2 _ NULL
         ("DIAGRAM" !L2!
            s1 _ #"DIAGRAM"
            <"from statement">
            param _ DSEL(#"STATEMENT")
            <"to statement">
            param2 _ DSEL(#"STATEMENT")
         %
         /"CELL" !L2!
            s1 _ #"CELL"
            <"or SUBCELL named">
            param _ LSEL(#"TEXT")
```

```
            <"reference coordinate">
            param2 _ DSEL ( #"BUG" )
            <"to">
            param3 _ DSEL ( #"BUG" )
         %
         /s1 _ figure <"at">
            param _ DSEL ( #"BUG" )  %cdsel(s1)%
            <"from">
            param2 _ DSEL ( #"BUG" )
            <"to">
            param3 _ DSEL ( #"BUG" )
         /s1 _ "GROUP" <"of figures in the window at">
            param _ DSEL ( #"BUG" )  %cdsel(s1)%
            <"and">
            s2 _ DSEL ( #"BUG" )
            <"from">
            param2 _ DSEL ( #"BUG" )
            <"to">
            param3 _ DSEL ( #"BUG" )
         )
      CONFIRM
      xgcopy(s1,param,s2,param2,param3)
      ;
   COMMAND zcreated =
      "CREATE" !L2!
      param2 _ NULL
         ("DIAGRAM"
            s1 _ #"DIAGRAM"
            <"at statement">
            param _ DSEL(#"STATEMENT")
         %
         /"CELL"
            s1 _ #"CELL"
            <"named">
            param _ LSEL(#"TEXT")
         /"SUBCELL"
            s1 _ #"SUBCELL"
            <"parent's name">
            param _ LSEL(#"TEXT")
            <"name">
            param2 _ LSEL(#"TEXT")
         %
         )
      CONFIRM
      xgorg(s1,param,param2)
      ;
   COMMAND zdeleted =
      "DELETE"
      s2 _ NULL
      (
         ( ("DIAGRAM" !L2!
            s1 _ #"DIAGRAM"
            <"at statement">
            param _ DSEL(#"STATEMENT")
         %
         /"CELL" !L2!
```

```
            s1 _ #"CELL"
            <"or SUBCELL named">
            param _ LSEL(#"TEXT")
         %
         /s1 _ figure <"at">
            param _ DSEL ( #"BUG" )
         /s1 _ "GROUP" <"of figures in the window at">
            param _ DSEL ( #"BUG" )  %cdsel(s1)%
            <"and">
            s2 _ DSEL ( #"BUG" )
         )
         CONFIRM
         xgerase(s1,param,s2) )
      /
         ( ("EDGE" <"at">
            s1 _ #"ERASE"
            param _ DSEL ( #"BUG" )  %cdsel(#"EDGE")%
         )
         param2 _ DSEL ( #"BUG" )
         xgvprt(s1,param,param2) )
      )
      ;
   COMMAND zextend =
      "XTEND"
         (s1 _ "VISIBLY" / s1 _ "INVISIBLY")
         <"the line at">
         param _ DSEL ( #"BUG" )
         <"to">
         param2 _ DSEL ( #"BUG" )
      CONFIRM
      xglong(s1,param,param2)
      ;
   COMMAND zinputd =
      "INPUT" !L2!
      <"SPL diagram from file">
      param _ LSEL(#"OLDFILENAME")
      <"block number">
      param2 _ LSEL(#"NUMBER")
      <"at statement">
      param3 _ DSEL(#"STATEMENT")
      CONFIRM
      xginpt(param, param2, param3)
      ;
   COMMAND zinsertd =
      "INSERT"
      s2 _ NULL
      param2 _ NULL
      param3 _ NULL
      (
         ( (param _ DSEL ( #"BUG" )
            s1 _ #"LINE"
            <"and">
            param2 _ DSEL ( #"BUG" )
         /"POINT" <"at">
            s1 _ #"POINT"
            param _ DSEL ( #"BUG" )
```

```
/"LINE" <"between">
   s1 _ #"UNCONSTRAINED"
   param _ DSEL ( #"BUG" )
   <"and">
   param2 _ DSEL ( #"BUG" )
/"HORIZONTAL" <"line starting at">
   s1 _ #"HORIZONTALLY"
   param _ DSEL ( #"BUG" )
   <"ending">
   param2 _ DSEL ( #"BUG" )
/"VERTICAL" <"line starting at">
   s1 _ #"VERTICALLY"
   param _ DSEL ( #"BUG" )
   <"ending">
   param2 _ DSEL ( #"BUG" )
/"RECTANGLE" <"corner at">
   s1 _ #"RECTANGLE"
   param _ DSEL ( #"BUG" )
   <"opposite at">
   param2 _ DSEL ( #"BUG" )
/"SQUARE" !L2! <"upper left corner at">
   s1 _ #"SQUARE"
   param _ DSEL ( #"BUG" )
   <"size">
   param2 _ DSEL ( #"BUG" )
/"DIAMOND" <"top at">
   s1 _ #"DIAMOND"
   param _ DSEL ( #"BUG" )
   <"bottom at">
   param2 _ DSEL ( #"BUG" )
   <"side at">
   param3 _ DSEL ( #"BUG" )
/"ARROWHEAD" <"pointing">
   s1 _ #"ARROWHEAD"
   (s2 _ bigdirection
      <"at">
      param _ DSEL ( #"BUG" )
   %/"ALONG" <"a line starting">
      s2 _ #"ALONG"
      param _ DSEL ( #"BUG" )
      <"ending">
      param2 _ DSEL ( #"BUG" )%
   )
/"TRIANGLE" <"pointing">
   s1 _ #"ITRIANGLE"
   (s2 _ bigdirection
      <"top">
      param _ DSEL ( #"BUG" )
      <"one base point at">
      param2 _ DSEL ( #"BUG" )
   %/"ALONG" <"a line starting">
      s2 _ #"ALONG"
      param _ DSEL ( #"BUG" )
      <"ending">
      param2 _ DSEL ( #"BUG" )
      <"one base point at">
```

```
                param3 _ DSEL ( #"BUG" )%
            )
         /"CIRCLE" <"center">
            s1 _ #"CIRCLE"
            param _ DSEL ( #"BUG" )
            (param2 _ DSEL ( #"BUG" )
               s2 _ #"TOUCHING"
            /"VERTICALLY" <"tangent to">
               s2 _ #"VERTICALLY"
               param2 _ DSEL ( #"BUG" )
            /"HORIZONTALLY" <"tangent to">
               s2 _ #"HORIZONTALLY"
               param2 _ DSEL ( #"BUG" )
            )
         )
         CONFIRM
         xgdraw(s1,s2,param,param2,param3) )
      /
         ("EDGE"  <"perpendicular to">
            (param _ DSEL ( #"BUG" )
               s1 _ #"PERPENDICULAR"
            /"CENTER"
               s1 _ #"CENTER"
               param _ DSEL ( #"BUG" )
            )
         param2 _ DSEL ( #"BUG" )
         xgvprt(s1,param,param2) )
      /
         ("STATEMENT" <"into the window at">
            param _ DSEL ( #"BUG" )
            <"and">
            param2 _ DSEL ( #"BUG" )
            param3 _ LSEL(#"STATEMENT")
         CONFIRM
         xgannote(param,param2,param3) )
      )
      ;
   COMMAND zlabel =
      "LABEL" <"diagram at">
         param _ DSEL ( #"BUG" )
         <"with">
         param2 _ LSEL(#"TEXT")
      CONFIRM
      xglabl(param,param2)
      ;
   COMMAND zmoved =
      "MOVE"
      param3 _ NULL
      s2 _ NULL
      (
         ( ("DIAGRAM" !L2!
            s1 _ #"DIAGRAM"
            <"from statement">
            param _ DSEL(#"STATEMENT")
            <"to statement">
            param2 _ DSEL(#"STATEMENT")
```

```
        %
        /"CELL" !L2!
           s1 _ #"CELL"
           <"or SUBCELL named">
           param _ LSEL(#"TEXT")
           <"reference coordinate">
           param2 _ DSEL ( #"BUG" )
           <"to">
           param3 _ DSEL ( #"BUG" )
        %
        /s1 _ figure <"at">
           param _ DSEL ( #"BUG" )
           <"from">
           param2 _ DSEL ( #"BUG" )
           <"to">
           param3 _ DSEL ( #"BUG" )
        /s1 _ "GROUP" <"of figures in the window at">
           param _ DSEL ( #"BUG" )  %cdsel(s1)%
           <"and">
           s2 _ DSEL ( #"BUG" )
           <"from">
           param2 _ DSEL ( #"BUG" )
           <"to">
           param3 _ DSEL ( #"BUG" )
        )
        CONFIRM
        xgmove(s1,param,s2,param2,param3) )
     /
        ( ("EDGE" <"from">
           s1 _ #"MOVE"
           param _ DSEL ( #"BUG" )  %cdsel(#"EDGE")%
           <"to">
           param2 _ DSEL ( #"BUG" )  %cdsel(#"EDGE")%
        )
        CONFIRM
        xgvprt(s1,param,param2) )
     )
     ;
  COMMAND zoutput =                                          6A50
     "OUTPUT" <"diagram at">
     param _ DSEL(#"STATEMENT")
     <"to COM file">
     namfil _ LSEL(#"NEWFILELINK")
     CONFIRM
     xgcout(namfil, param)
     ;
  COMMAND zportray =
     "PORTRAY" <"diagram at statement">
     param _ DSEL(#"STATEMENT")
     <"in graphics window">
     param2 _ DSEL ( #"BUG" )
     xgport(param,param2)
     ;
  COMMAND zreset =
     "RESET"
        s1 _ "DRAWSPECS"
```

```
      CONFIRM
      xgreset(s1)
      ;
   COMMAND zsetd =
      "SET"
         param2 _ NULL
         (s1 _ "MARGIN"
            (param _ margin param2 _ DSEL ( #"BUG" )
            /param _ "OFF" param2 _ NULL
            /param _ "ON" !L2! param2 _ NULL)
         /s1 _ "VIEWGRAPH" <"mode on/off">
            (param _ "OFF" / param _ "ON" !L2!)
         /s1 _ "JUSTIFICATION"
            param _ justify
         /s1 _ "ARROWHEAD" <"height">
            param _ LSEL(#"TEXT")
            <"width"> param2 _ LSEL(#"TEXT")
         /s1 _ "GRID" <"size">
            param _ LSEL(#"TEXT")
         /s1 _ "CHARACTER" <"height">
            param _ LSEL(#"TEXT")
         /s1 _ "FONT"
            param _ font
         /s1 _ "EMPHASIS"
            param _ tstyle
         / "LINE"
            (s1 _ "TYPE" <"to"> param _ ltype
            /s1 _ "STYLE" <"to"> param _ lstyle)
         /s1 _ "TERMINAL" <"type">
            (param _ "T2"
            /param _ "T4" !L2!)
         )
      CONFIRM
      xgset(s1,param,param2)
      ;
   COMMAND zshowd =
      "SHOW" !L2!
         (s1 _ "GRID" <"in the window">
            param _ DSEL ( #"BUG" )
            <"other corner">
            param2 _  DSEL ( #"BUG" )
            CONFIRM
         /s1 _ "DRAWSPECS" <"for window">
            param _ DSEL ( #"BUG" )
            param2 _ NULL
         /s1 _ "SPECIFIC" <"drawspecs for">
            param _ figure <"at">
            param2 _ DSEL ( #"BUG" )
            CONFIRM
         )
      xgshow(s1,param,param2)
      ;
   COMMAND zshowf =
      "."
         (<"Show Grid in the window">
            s1 _ #"GRID"
```

```
            param _ DSEL ( #"BUG" )
            <"other corner">
            param2 _  DSEL ( #"BUG" )
         )
      CONFIRM
      xgshow(s1,param,param2)
      ;
   COMMAND ztest =
      "TEST" !L2! <"*** Configuration Checkout ***"> CONFIRM
      xgtest();
   COMMAND zupdate =
      "UPDATE" <"graphics display"> CONFIRM xgupda();
   COMMAND zzoom =
      "ZOOM" <"or move image">
         (param _ DSEL ( #"BUG" )
            <"to">
            (param2 _ DSEL ( #"BUG" )
               s1 _ #"MOVE"
            /"CENTER"
               s1 _ #"CENTER"
               param2 _ NULL
            )
         /"OUT"
            s1 _ #"OUT"
            param _ NULL
            param2 _ NULL
         /"FULL" <"size of the margins">
            s1 _ #"FULL"
            param _ NULL
            param2 _ NULL
         /"TO" <"window">
            s1 _ #"TO"
            <"one corner">
            param _ DSEL ( #"BUG" )
            <"other corner">
            param2 _ DSEL ( #"BUG" )
         /"IN"
            s1 _ #"IN"
            param2 _ NULL
            param _ NULL
         )
      CONFIRM
      xgwdom(s1,param,param2)
      ;
   %
      COMMAND zannote =
         "ANNOTATE" <"the place at">
         param _ DSEL ( #"BUG" )
         <"and">
         param2 _ DSEL ( #"BUG" )
         param3 _ LSEL(#"TEXT")
         CONFIRM
         xgannote(param,param2,param3)
         ;
      COMMAND zassociate =
         "ASSOCIATE" !L2! <"graphics viewport">
```

```
            param _ DSEL ( #"BUG" )
            <"with text place">
            param2 _ DSEL ( #"BUG" )
            s1 _ #"ASSOCIATE"
            CONFIRM
            xgvprt(s1,param,param2)
            ;
         COMMAND zdissociate =
            "DISSOCIATE" !L2! <"graphics viewport">
            param _ DSEL ( #"BUG" )
            s1 _ #"DISSOCIATE"
            param2 _ NULL
            CONFIRM
            xgvprt(s1,param,param2)
            ;
         COMMAND zrefer =
            "REFER" <"to cell or subcell named">
            param _ LSEL(#"TEXT")
            CONFIRM
            xgref(param)
            ;
         COMMAND ztransform =
            "TRANSFORM"
               (s1 _ figure <"at">
                  param _ DSEL ( #"BUG" )
                  <"by">
                  param2 _ NULL
                  s2 _ grotate <"counterclockwise quadrant
                  rotations">
               /s1 _ "GROUP" <"of figures in the window at">
                  param _ DSEL ( #"BUG" )
                  <"and">
                  param2 _ DSEL ( #"BUG" )
                  s2 _ grotate <"counterclockwise quadrant
                  rotations">
               )
            CONFIRM
            xgtran(s1,s2,param,param2)
            ;
      %
      INCLUDE <nine, nls-grammar, commands !universal>          6A57
      END.
   FINISH
%parse functions%                                                 6B
   FILE graphicspfc % <arcsubsys,x110,>  <arcsubsys,1109,> to
   <relnine, graphics.pfc, >%
   % Declarations %
      (gbadcoords) = $"Place cursor mark in graphics display";   6B2A
   % Code %
      (gpgetcoords) % get coordinates on graphics display for DSEL %
      COROUTINE (type, inst, accumulator REF);                  6B3A
         % Return coordinate list: x-coord, y-coord, window id.
         Mask coordinates to 10 bits %
         LOCAL gcoord = 2000B, acvalue = accumulator, cmlvar REF, x,
         y, windid;
         %--------------------%
```

```
PORT ENTRY EXIT PCALL;
accumlator _ acvalue;
IF lastx < gcoord THEN
   err(gbadcoords);
windid _ dpycall( ixywindow, 2, lastx, lasty : x, y );
dpycall( imarkfcn, 8, domark, mkpos, windid, 0, 0, 0,
lastx, lasty);
&cmlvar _ getblk( 5, $freespace);
cmlvar.vtype _ listtype;
cmlvar.vlength _ 5;
cmlvar.vmarks _ 1; %made one mark on display%
cmlvar[1] _ 3; %three things in list%
cmlvar[2] _ makecmlint(x .A 1777B);
cmlvar[3] _ makecmlint(y .A 1777B);
cmlvar[4] _ makecmlint(windid);
accumlator _ &cmlvar;
PCALL( dosuc);
END.%%
```

```
   FINISH
%be%                                                                  6C
   FILE graphics % in NLS 8.5  (arcsubsys,x110,) in NLS 9
   (arcsubsys,1109,)   (relnine, graphics.subsys,) %
      ALLOW!
      % Declarations %                                                6C1B
         % Disptach table %                                           6C1B1
            (graphics) EXTERNAL _ (                                   6C1B1A
               $"XGANNOTE", $xgannote,
               $"XGCHNG", $xgchng,
               $"XGCOPY", $xgcopy,
               $"XGCOUT", $xgcout,
               $"XGDRAW", $xgdraw,
               $"XGERASE", $xgerase,
               $"XGINIT", $xginit,
               $"XGINPT", $xginpt,
               $"XGLABL", $xglabl,
               $"XGLONG", $xglong,
               $"XGMOVE", $xgmove,
               $"XGORG", $xgorg,
               $"XGMOVE", $xgmove,
               $"XGPORT", $xgport,
               $"XGREF", $xgref,
               $"XGRENTR", $xgrentr,
               $"XGRESET", $xgreset,
               $"XGSET", $xgset,
               $"XGSHOW", $xgshow,
               $"XGTERM", $xgterm,
               $"XGTEST", $xgtest,
               $"XGTRAN", $xgtran,
               $"XGUPDA", $xgupda,
               $"XGVPRT", $xgvprt,
               $"XGWDOW", $xgwdom,
               0,0);  %end of table%
         %Registers%
            (m11) EXTERNAL ADDRESS = 0;                               6C1B2A
            (m12) EXTERNAL ADDRESS = 3;                               6C1B2B
            (m21) EXTERNAL ADDRESS = 1;                               6C1B2C
            (m22) EXTERNAL ADDRESS = 4;                               6C1B2D
            (m31) EXTERNAL ADDRESS = 2;                               6C1B2E
            (m32) EXTERNAL ADDRESS = 5;                               6C1B2F
         %State control for the subsystem%
            %controlling tables and their addresses%
               (gprwndw)  EXTERNAL =  0;                              6C1B3A1
                  %address of wa - initial graphics window%
               (gdabas) EXTERNAL = 0;                                 6C1B3A2
                  %base address - graphics display areas%
               (gdaphm) EXTERNAL = 0;                                 6C1B3A3
                  %phantom gda has global system variables%
               (gwautl) EXTERNAL = 3;  %number of wa's%               6C1B3A4
               (gwaut ) EXTERNAL = (0, 0, 0);  %use table for wa's%
                                                                      6C1B3A5
               (attach) EXTERNAL = 0;   %address of the attaching
               table%                                                 6C1B3A6
               (atvisible) EXTERNAL = 0;                              6C1B3A7
```

```
      %attaching point under construction visible%
   (atch) EXTERNAL = 0;    %attaching point under
   construction%                                          6C1B3A8
   (gunitb) EXTERNAL = 0;  %unit table in use%    6C1B3A9
   (gbiunt) EXTERNAL = 0;  %built in unit table% 6C1B3A10
   (gbuabb) EXTERNAL = 0;                        6C1B3A11
      %basic unit abbreviation (string address)%
   (gatbox) EXTERNAL = 100;  %radius of attaching box%
                                                 6C1B3A12
%initial state information%
   (setmtop   ) EXTERNAL = 11000;  %initial extent
   boundary.%                                             6C1B3B1
   (setmbottom) EXTERNAL = 0;  %8 1/2 X 11 inches.%
                                                          6C1B3B2
      %1000 incr./in.%
   (setmleft  ) EXTERNAL = 0;                             6C1B3B3
   (setmright ) EXTERNAL = 8500;                          6C1B3B4
   (logof) EXTERNAL = 0;                                  6C1B3B5
   (gmgnfl) EXTERNAL = 1; %margin display flag%   6C1B3B6
%initial unit definitions relative to inch%
   (iunit) EXTERNAL STRING =   %default basic unit is
   first%                                                 6C1B3C1
      "in,1,cm,.39,mm,.04,dm,3.94,m,39.37,ft,12,yd,36,mi
      ,63360,pt,.013837,";
%default values%
   (dfltcs) EXTERNAL STRING = "12pt";  %character size%
                                                          6C1B3D1
   (dfltah) EXTERNAL STRING = "8pt";  %arrow head
   height%                                                6C1B3D2
   (dfltaw) EXTERNAL STRING = "6pt";  %arrowhead width%
                                                          6C1B3D3
   (dfltr) EXTERNAL  = 212764000000B;  %resolution
   default%                                               6C1B3D4
      %(floating 1000)%
   (dfltg) EXTERNAL  = 207620000000B;  %grid default%
                                                          6C1B3D5
      %(floating 100)%
   (dfltj) EXTERNAL  = 1;  %justification - left% 6C1B3D6
   (dfltb) EXTERNAL  = 60B;  %text window border% 6C1B3D7
   (dfltlt) EXTERNAL  = 1;  %line type - solid%   6C1B3D8
   (dfltls) EXTERNAL  = 3;  %line style - regular%
                                                          6C1B3D9
   (dfltcf) EXTERNAL  = 6;  %character font -
   TimesRoman%                                           6C1B3D10
%boldness increase data for 4014/4012%
   (gblrrc) EXTERNAL  = 1; %number of redrawings (*2)%
                                                          6C1B3E1
   (gxblrr) EXTERNAL  = 0; %x displacement for
   redrawing%                                             6C1B3E2
   (gyblrr) EXTERNAL  = 0; %y displacement for
   redrawing%                                             6C1B3E3
   (gtblrr) EXTERNAL  =    %table of displacements%
                                                          6C1B3E4
      (0, 0, 5, 0, 5, 5, 0, 5);
%device type%
   (gdevtyp) EXTERNAL  = 2;  %graphics device type%
```

```
                                                                6C1B3F1
        (t4012) EXTERNAL  = 1;  %Tektronix 401;%    6C1B3F1A
        (t4014) EXTERNAL  = 2;  %Tektronix 4014%    6C1B3F1B
        (vcom ) EXTERNAL  = 3;  %virtual COM device%
                                                    6C1B3F1C
        (singer) EXTERNAL  = 4;  %Singer 6000%      6C1B3F1D
        (comp80) EXTERNAL  = 5;  %III Comp 80%      6C1B3F1E
%line style table%
%buffering data%
   (clast ) EXTERNAL = 7777777777777B;              6C1B3H1
   (tekmode) EXTERNAL = 1;                          6C1B3H2
   (graphm) EXTERNAL = 2;                           6C1B3H3
   (alpham) EXTERNAL = 1;                           6C1B3H4
   (sendp) EXTERNAL;                                6C1B3H5
   (sendl) EXTERNAL;                                6C1B3H6
%COM output data%
   (adout1) EXTERNAL = 0;  %addr. of the op's byte
   output routine%                                  6C1B3I1
   (comlsg) EXTERNAL = 1;  %virtual COM line segment
   code%                                            6C1B3I2
   (compag) EXTERNAL = 2;  %pagination code%        6C1B3I3
   (comfig) EXTERNAL = 3;  %insert figure%          6C1B3I4
   (comeof) EXTERNAL = 4;  %end of document code%   6C1B3I5
   (comln ) EXTERNAL = 5;  %linework code%          6C1B3I6
   (comnul) EXTERNAL = 177B;  %odd but true%        6C1B3I7
   (comfnt) EXTERNAL = 6;  %default com font times
   roman%                                           6C1B3I8
   (comtyp) EXTERNAL = 0;  %default type style%     6C1B3I9
   (comch ) EXTERNAL = 0;  %height of the character for
   com%                                             6C1B3I10
   (comxlfb) EXTERNAL = 0;  %default figure boundaries
   for COM%                                         6C1B3I11
   (comxrfb) EXTERNAL = 8500;                       6C1B3I12
   (comytfb) EXTERNAL = 0;                          6C1B3I13
   (comybfb) EXTERNAL = 11000;                      6C1B3I14
   (comjfn) EXTERNAL = 0;                           6C1B3I15
%SPL data%
   (sple01) EXTERNAL = 1;     %margin error%        6C1B3J1
   (sple02) EXTERNAL = 2;     %unknown command%     6C1B3J2
   (sple03) EXTERNAL = 4;     %line error%          6C1B3J3
   (sple04) EXTERNAL = 10B;   %text error%          6C1B3J4
   (sple05) EXTERNAL = 20B;   %keyword unknown%     6C1B3J5
   (sple06) EXTERNAL = 40B;   %keyword value error%
                                                    6C1B3J6
   (sple07) EXTERNAL = 100B; %data range error%     6C1B3J7
   (spleol) EXTERNAL = 0;  %end of line flag%       6C1B3J8
   (spljfn) EXTERNAL = 0;                           6C1B3J9
%constants and data%
   (glastxy) EXTERNAL;  %Last coordinates bugged%   6C1B3K1
%set and change fields%
   (jstfct) EXTERNAL CONSTANT = 1;  %justification%
                                                    6C1B3L1
   (chsize) EXTERNAL CONSTANT = 2;  %character size%
                                                    6C1B3L2
   (font) EXTERNAL CONSTANT = 3;  %font%            6C1B3L3
   (ltype) EXTERNAL CONSTANT = 4;  %line type%      6C1B3L4
```

```
      (lstyle) EXTERNAL CONSTANT = 5;  %line style%  6C1B3L5
   %font codes%
      (fntc) EXTERNAL CONSTANT = 0;  %Courier%       6C1B3M1
      (fntd) EXTERNAL CONSTANT = 1;  %Directory%     6C1B3M2
      (fntf) EXTERNAL CONSTANT = 2;  %Film%          6C1B3M3
      (fnto) EXTERNAL CONSTANT = 3;  %OCRB%          6C1B3M4
      (fntnm) EXTERNAL CONSTANT = 4;  %NMAMIcrofont% 6C1B3M5
      (fntn) EXTERNAL CONSTANT = 5;  %NewsGothic%    6C1B3M6
      (fntt) EXTERNAL CONSTANT = 6;  %TimesRoman%    6C1B3M7
   %vector line types%
      (vcsolid) EXTERNAL CONSTANT = 1;               6C1B3N1
      (vcdot) EXTERNAL CONSTANT = 2;                 6C1B3N2
      (vcddash) EXTERNAL CONSTANT = 3;               6C1B3N3
      (vcsdash) EXTERNAL CONSTANT = 4;               6C1B3N4
      (vcldash) EXTERNAL CONSTANT = 5;               6C1B3N5
   %line styles%
      (lsextra) EXTERNAL CONSTANT = 1;  %extra fine% 6C1B3O1
      (lsfine) EXTERNAL CONSTANT = 2;  %fine%        6C1B3O2
      (lsregular) EXTERNAL CONSTANT = 3;  %regular%  6C1B3O3
      (lsbold) EXTERNAL CONSTANT = 4;  %bold%        6C1B3O4
      (lsvbold) EXTERNAL CONSTANT = 5;  %very bold%  6C1B3O5
   %intensity, broadness table%
      (lstabl) EXTERNAL =  %intensity, broadness%    6C1B3P1
         (8, 0,  %extra fine%
         8, 3,  %fine%
         8, 5,  %regular%
         8, 8,  %bold%
         8, 11);  %very bold%
      (lstsiz) EXTERNAL CONSTANT = 5;  %line style table
      size%                                          6C1B3P2
   %manager mode codes%
      (gcreate) EXTERNAL CONSTANT = 1;               6C1B3Q1
      (gdelete) EXTERNAL CONSTANT = 2;               6C1B3Q2
      (gcopyf ) EXTERNAL CONSTANT = 3;               6C1B3Q3
   %floating point constants%
      (fhalf ) EXTERNAL CONSTANT = 2004000000000B;   6C1B3R1
      (fone ) EXTERNAL CONSTANT = 201400000000B;     6C1B3R2
      (ftwo ) EXTERNAL CONSTANT = 202400000000B;     6C1B3R3
      (ffour) EXTERNAL CONSTANT = 203400000000B;     6C1B3R4
      (ften) EXTERNAL CONSTANT = 204500000000B;      6C1B3R5
      (fthou) EXTERNAL CONSTANT = 212764000000B;     6C1B3R6
      (fmthou) EXTERNAL CONSTANT = 5650140000000B;   6C1B3R7
      (fpoint) EXTERNAL CONSTANT = 172705322256B;
      %.013837%                                      6C1B3R8
%Tektronix constants%
   %TEKTRONIX control characters%
      (tekgm) EXTERNAL CONSTANT = 29;  %enter graphics
      mode%                                          6C1B4A1
      (tekam) EXTERNAL CONSTANT = 31;  %enter alpha mode%
                                                     6C1B4A2
      (hybyte) EXTERNAL CONSTANT =  40B; %flag bits for a
      hi y byte%                                     6C1B4A3
      (lybyte) EXTERNAL CONSTANT = 140B; %low y byte%
                                                     6C1B4A4
      (hxbyte) EXTERNAL CONSTANT =  40B; %hi x byte% 6C1B4A5
      (lxbyte) EXTERNAL CONSTANT = 100B; %low x byte%
```

```
                                                                6C1B4A6
        (tekpad) EXTERNAL CONSTANT = 31B; %padding%             6C1B4A7
           %to insure completion of last vector%
        (tekesc) EXTERNAL CONSTANT =  33B; %escape character%
                                                                6C1B4A8
        (teksty) EXTERNAL CONSTANT = 140B; %line style
        character%                                              6C1B4A9
        (tekclr) EXTERNAL CONSTANT =  12;  %clear screen%
                                                               6C1B4A10
     %TEKTRONIX state values%
        (gtekln) EXTERNAL =0;  %line style state%               6C1B4B1
        (gtekch) EXTERNAL =0;  %character size select state%
                                                                6C1B4B2
        (tekcsize) EXTERNAL = (                                 6C1B4B3
           56, 88, 70B,  %large size%
           51, 82, 71B,
           34, 53, 72B,
           31, 48, 73B); %small  size%
     %Display boundary values%
        (scntop ) EXTERNAL = 3210;                              6C1B4C1
        (scnbot ) EXTERNAL = 0;                                 6C1B4C2
        (scnleft) EXTERNAL = 0;                                 6C1B4C3
        (scnright) EXTERNAL = 4095;                             6C1B4C4
        (scnvtop) EXTERNAL = 2932;                              6C1B4C5
  %Device status data%
     (gcurvc) EXTERNAL = 0;  %vector type%                       6C1B5A
     (gcurin) EXTERNAL = 0;  %intensity%                         6C1B5B
     (gcurbd) EXTERNAL = 0;  %broadness%                         6C1B5C
     (gcurtf) EXTERNAL = (0,0,0,0,0);  %text format record%
                                                                 6C1B5D
     (gcurch) EXTERNAL = 0;  %character height in screen
     coords%                                                     6C1B5E
     (gcurts) EXTERNAL = 0;                                      6C1B5F
        %TRUE it the character is too small to be displayed%
  %Lineprocessor protocol constants%
     %graphics protocol code for the lineprocessor%
        (gpesc) EXTERNAL CONSTANT = 33B;  %protocol escape
        code%                                                    6C1B6A1
        (gpopen) EXTERNAL CONSTANT = 53B;                        6C1B6A2
           %open the port for access to the tektronix%
        (gpclos) EXTERNAL CONSTANT = 54B;                        6C1B6A3
           %close the port and allow tracking%
        (gpstr) EXTERNAL CONSTANT = 52B;                         6C1B6A4
           %character string directed to the display%
        (gpdev) EXTERNAL CONSTANT = 40B;                         6C1B6A5
           %device code - not really used but in the string -
           %
        (gpbias) EXTERNAL CONSTANT = 40B;  %bias added to
        string count%                                            6C1B6A6
  %Viewport constants%
     %which edge%
        (ledge) EXTERNAL CONSTANT = 83;  %left edge%    6C1B7A1
        (redge) EXTERNAL CONSTANT = 84;  %right edge%   6C1B7A2
        (tedge) EXTERNAL CONSTANT = 94;  %top edge%     6C1B7A3
        (bedge) EXTERNAL CONSTANT = 95;  %bottom edge%  6C1B7A4
  %Parsing input constants%
```

```
   (ascii0) EXTERNAL CONSTANT = 60B;  %ascii value of
   character zero%                                          6C1B8A
   (gtext) EXTERNAL STRING [15];  %input text%              6C1B8B
%FE commands data%                                          6C1B9
   (maxfei) EXTERNAL CONSTANT = 6;  %max. instr. can send
   to fe%                                                   6C1B9A
   (gcommands) EXTERNAL;  %addr. of list for batch command%
   (garglist) EXTERNAL;  %addr. of list for call on FE%
   (gshwst) EXTERNAL;  %addr. of string for SHOW command%
   (sendbf) EXTERNAL ;  %addr. of string for write lit.
   command%                                                 6C1B9E
   (gbfsize) EXTERNAL CONSTANT = 1200;  %FE buffer size%
                                                            6C1B9F
   (garglsize) CONSTANT = 10;  %size of 'garglist'%
%Command word token values%                                 6C1B10
   (cwannotation) CONSTANT = 29;  %STATEMENT in grammar%
%Test declarations%
   (tstgda) EXTERNAL = (                                    6C1B11A
      0,
      0,
      0077777000000B,
      0000000005564B,
      0,0,0,0,0,0,0,0,0,
      0,0,0,0,  %window coordinates%
      0,0,0,0,0,0,  %viewing transformation%
      0,0,0,0,0,0,  %pointing transformation%
      0,0,  %margins%
      102000060B,   %text format%
      60B,
      400B,
      2761B,   %line format%
      0,    %coordinates of the last line for continue%
      60000024B,   %arrowhead height%
      212764000000B,  %page units per problem unit
      (floating 1000)%
      207620000000B);  %page units per grid increment
      (floating 100)%
   %aspect ratio character h/w = 1.384%
      (aspctr) EXTERNAL CONSTANT = 201542233513B;   6C1B11B1
%Constants for fbctl - should be in NLS constant file%
                                                            6C1B12
   (typecalit) EXTERNAL CONSTANT = 11;      % wait for CA
   after printing literal %                                 6C1B12A
   (fbaddlit) EXTERNAL CONSTANT = 12;       % add string to
   literal feedback area %                                  6C1B12B
   (addcalit) EXTERNAL CONSTANT = 13;       % wait for CA %
                                                            6C1B12C
   (typenulllit) EXTERNAL CONSTANT = 14;    % type null
   literal string %                                         6C1B12D
   (fbendlit) EXTERNAL CONSTANT = 15;       % type lit and
   wait for input %                                         6C1B12E
%Record definitions%
   (gdhrec) RECORD  %Diagram header record%                 6C1B13A
      gmargleft[18],   %extent boundary%
      gmargright[18],
      gmargtop[18],
```

```
      gmargbottom[18],
      gdhres[36],  %resolution (increments / unit)%
      gdhdum[36];  %First of 50 free words%
      %Associated constants%
         (gdhsiz) EXTERNAL CONSTANT = 53;  %diagram header
         size%                                          6C1B13A7A
   (gchrec) RECORD  %Cell header record%                  6C1B13B
      gctype[4],   %cell type%
      gdum1[32],
      gcorg[36];  %x and y origin - abs page coords%
      %Associated constants%
         (gchnfst) EXTERNAL CONSTANT = 2;  %index to first
         name word%                                     6C1B13B4A
         (gncell ) EXTERNAL CONSTANT = 1;  %named cell
         code%                                          6C1B13B4B
         (gpcell ) EXTERNAL CONSTANT = 2;  %phantom cell
         code%                                          6C1B13B4C
         (gccell ) EXTERNAL CONSTANT = 3;  %construction
         cell%                                          6C1B13B4D
   (gtfrec) RECORD  %Graphics text format record%       6C1B13C
      gtfcds[36],  %upper left coordinates%
      gtfwdt[18],  %width%
      gtfhgt[18],  %height%
      gtfcw [18],  %character width%
      gtfch [18],  %character height%
      gtfbdr[18],  %border between window and text%
      gtffnt[18],  %font%
      gtfemp[ 8],  %emphasis byte%
      gtfjus[ 4];  %justification%
   (gwautr) RECORD  %Work area use table entry format%
                                                        6C1B13D
      gwaaddr[18],   %core address of the actual work area%
      gwaage[18];    %number of writes from this wa%
   (gwarec) RECORD  %Work area header%                    6C1B13E
      gwacstid[36],  %STID of the cell inferior tree%
      gwalstid[36],  %STID of the line work property block
      this wa%
      gwaclgt[18],   %current length%
      gwamlgt[18];   %maximum length%
      %Associated constants%
         (gwahdl) EXTERNAL CONSTANT = 3;    %wa header
         length%                                        6C1B13E5A
         (gwaml ) EXTERNAL CONSTANT = 250;  %wa maximum
         length - maximum data area of linework property%
                                                        6C1B13E5B
   (gpatch) RECORD  %Attaching table entry%               6C1B13F
      gatcd[36],    %x and y coordinates%
      gatstid[36],  %line work node address%
      gatdsp[9],    %displacement to the graphics
      instruction%
      gatop[8],     %graphics instruction op code%
      gatnew[1];    %used by move to indicate modification%
      %attaching table pointers and data%
         (gatsiz) EXTERNAL CONSTANT = 100;          6C1B13F6A
            %initial size of the attaching table%
         (gatexp) EXTERNAL CONSTANT = 50;   %expansion
```

```
            size%                                                6C1B13F6B
            (gattel) EXTERNAL CONSTANT = 3;     %entry length%
                                                                 6C1B13F6C
            (gatml) EXTERNAL CONSTANT = 0;      %maximum length
            index%                                               6C1B13F6D
            (gatcl) EXTERNAL CONSTANT = 1;      %current length
            index%                                               6C1B13F6E
            (gatgl) EXTERNAL CONSTANT = 2;      %number of
            deleted entries index%                               6C1B13F6F
            (gatptr) EXTERNAL CONSTANT = 3;     %pointer to
            next entry during group command%                     6C1B13F6G
            (gatfst) EXTERNAL CONSTANT = 4;     %index of first
            entry in the table%                                  6C1B13F6H
         (gutent) RECORD  %Unit table entry%                       6C1B13G
            gutshd[36],  %abbreviation string header%
            gutstx[36],  %abbreviation string text%
            gutpsi[36];  %number of page space icrements
            (floating point)%
            %unit table constants%
               (guntel) EXTERNAL CONSTANT = 3;  %unit table entry
               length%                                           6C1B13G4A
               (guntal) EXTERNAL CONSTANT = 5;  %abbreviation
               length (maximum)%                                 6C1B13G4B
               (guntsz) EXTERNAL CONSTANT = 20;  %unit table
               size%                                             6C1B13G4C
               (guntcl) EXTERNAL = 0;  %unit table current
               length%                                           6C1B13G4D
         (gphins) RECORD  %Graphic instruction format%         6C1B13H
            gphops[8],  %instruction opcode (type)%
            gphlgt[8],  %instruction length in words including
            the first%
            gphmor[1],  %true if flags cntinue to the next word%
            gphatt[1],  %true if attachable (more prominant for
            pointing)%
            gphvec[4],  %vector type%
               %
               vcsolid   = solid vector
               vcdotted  = dotted vector
               vcdashed  = dashed vector
               %
            gphint[4],  %intensity of painted linework%
               %
                0 = blank
               15 = full bright
               %
            gphbrd[4],  %broadness of painted linework%
               %
                0 = narrowest
               15 = broadest
               %
            gphnll[6];  %free%

            %graphics instruction op codes%
               (gpenup) EXTERNAL CONSTANT = 400000400000B;  %pen
               up in long lines%                          6C1B13H10A
               (figure) EXTERNAL CONSTANT = 1;            6C1B13H10B
```

```
        (rectangle) EXTERNAL CONSTANT = 2;          6C1B13H10C
        (line) EXTERNAL CONSTANT = 3;               6C1B13H10D
        (point) EXTERNAL CONSTANT = 5;              6C1B13H10E
        (circle) EXTERNAL CONSTANT = 6;             6C1B13H10F
        (diamond) EXTERNAL CONSTANT = 7;            6C1B13H10G
        (itriangle) EXTERNAL CONSTANT = 8;          6C1B13H10H
        (arrowhead) EXTERNAL CONSTANT = 9;          6C1B13H10I
        (arc) EXTERNAL CONSTANT = 10;               6C1B13H10J
        (ellipse) EXTERNAL CONSTANT = 11;           6C1B13H10K
        (earc) EXTERNAL CONSTANT = 12;              6C1B13H10L
        (annotation) EXTERNAL CONSTANT = 14;        6C1B13H10M
      %direction codes%
        (east   ) EXTERNAL CONSTANT = 0;            6C1B13H11A
        (noreast) EXTERNAL CONSTANT = 1;            6C1B13H11B
        (north  ) EXTERNAL CONSTANT = 2;            6C1B13H11C
        (norwest) EXTERNAL CONSTANT = 3;            6C1B13H11D
        (west   ) EXTERNAL CONSTANT = 4;            6C1B13H11E
        (souwest) EXTERNAL CONSTANT = 5;            6C1B13H11F
        (south  ) EXTERNAL CONSTANT = 6;            6C1B13H11G
        (soueast) EXTERNAL CONSTANT = 7;            6C1B13H11H

    (gdisplayarea) RECORD   %Graphics display area%   6C1B13I

      gdanxt[18],       %next gda in list%
      gdapvs[18],       %previous gda in list%

      gtype[36],        %type and flags in this word ***%

      gleft[18],        %left boundary -viewport -screen cds%
      gright[18],       %right boundary%
      gtop[18],         %top boundary%
      gbottom[18],      %bottom boundary%

      gdstid[36],       %STID diagram%
      gtstid[36],       %STID of the diagram inferior tree%
      gcstid[36],       %STID cell%
      glstid[36],       %STID open linework inferior tree%
      gorg[36],         %effective x and y origin of the open
      cell%

      gvspec[36],       %first viewspec word%
      gvspc2[36],       %second viewspec word%
      gdspec[36],       %first drawspec word%

      gwleft[36],       %left boundary -window -floating page
      cds%
      gwright[36],      %right boundary%
      gwtop[36],        %top boundary%
      gwbottom[36],     %bottom boundary%

      gvtrn[36],        %viewing transformation - page to
      screen%
      gvtrn2[36],       % - six floating point words - %
      gvtrn3[36],
      gvtrn4[36],
      gvtrn5[36],
```

```
        gvtrn6[36],

        gptrn[36],      %pointing transformation - screen to
        page%
        gptrn2[36],      % - six floating point words - %
        gptrn3[36],
        gptrn4[36],
        gptrn5[36],
        gptrn6[36],

        gmleft[18],     %Diagram margin - page cds - integer%
        gmright[18],
        gmtop[18],
        gmbottom[18],

                        %Graphics text format%
        gdacw [18],     %character width%
        gdach [18],     %character height%
        gdabdr[18],     %border between window and text%
        gdafnt[18],     %font%
        gdaemp[ 8],     %emphasis byte%
        gdajus[ 4],     %justification%
        gdadm2[24],     %free%

        gdalvec[4],     %vector type%
        gdalint[4],     %intensity%
        gdalbrd[4],     %broadness%
        gdadm3[24],     %free%

        gdalcd[36],     %last coordinate of a line%

        gdaawdt[18],     %arrowhead width%
        gdaahgt[18],     %arrowhead height%
                         %Page coordinate system relationships%
        gdares[36],      %Number of page space units per problem
        unit%
        gdagrd[36];      %Number of page space units per grid
        increment%

        %Display area constants, pointers, and indices%
           (gdalgt) EXTERNAL CONSTANT = 38;   %length%
                                                    6C1B13I68A
           (gdavx) EXTERNAL CONSTANT = 16;    %index of
           viewing transformation%                  6C1B13I68B
           (gdawdo) EXTERNAL CONSTANT = 12;   %index of the
           viewing window%                          6C1B13I68C
           (gdapx) EXTERNAL CONSTANT = 22;    %index of the
           pointing transformation%                 6C1B13I68D
           (gdamgn) EXTERNAL CONSTANT = 28;   %index of the
           margin boundary%                         6C1B13I68E
           (gdagtf) EXTERNAL CONSTANT = 30;   %index of the
           text format%                             6C1B13I68F
      (tekrec) RECORD xb[2], lo[5], hi[5];             6C1B13J
      (comcds) RECORD b7low[7], b7high[7];             6C1B13K
      (cdsrec) RECORD xc[18], yc[18];                  6C1B13L
      (cdsdwh) RECORD hgt[16], wdt[16], dir[4];        6C1B13M
```

```
    (gemphasis) RECORD %instances of emphasis bytes in text
    format%                                                  6C1B13N
       gempb[2],  %boldness 0, 1, or 2%
       gemps[1],  %TRUE for slanted%
       gempu[1],  %TRUE for underlined%
       gempm[1];  %TRUE for monospacing%
%FINISH.%%% % <NUL><NUL><^P><^U>gf data for split version
```

```
      of subsystem %
   %Parsefunctions%
   %Execution routines%
      (xgannote) PROCEDURE (bug1 REF LIST,bug2 REF LIST,address
      REF LIST);                                                    6C1D1
         %caption diagrams%
         LOCAL scd1, scd2, text REF, stid, gda, pcd1, pcd2, ju,
         il, w, l;
         LOCAL TEXT POINTER tp1, tp2;
         REF gda;
         %---------------%
         % convert from fe argument format %
            scd1 _ gcnvbug( &bug1 );
            scd2 _ gcnvbug ( &bug2 );
            &text _ ELEM #address#[tppair];  %addr. of text
            pointers%
         &gda _ gdacds(scd1);
         gckemp(&gda);
         gckopn(&gda);
         tp1 _ text;
         tp1[1] _ text[1];
         tp2 _ text [2];
         tp2[1] _ text[3];
         pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
         pcd2 _ gridxy(&gda, scd2.xc, scd2.yc);
         ju _ maxse((pcd1.yc), (pcd2.yc));
         il _ minse((pcd1.xc), (pcd2.xc));
         w _ absdif(pcd1.xc, pcd2.xc);
         l _ absdif(pcd1.yc, pcd2.yc);
         stid _ gbldtn(&gda, gpack(il, ju), w, l, $tp1, $tp2);
         gpsopn();
         gmvtn(&gda, stid, gda.gorg, FALSE);
         gpscls();
         RETURN;
         END.
      (xgchng) PROCEDURE (ent REF LIST,ent2 REF LIST,ent3 REF
      LIST,bug1 REF LIST);                                          6C1D2
         %Change graphics instruction and text format parameters%
         LOCAL text REF, param, entity, entty2, scd, fig, pvalue;
         %---------------%
         % Convert from FE argument format %
            scd _ gcnvbug( &bug1 );
            entity _ ELEM #ent#[cwtype];
            entty2 _ ELEM #ent2#[cwtype];
         CASE entity OF
            =10:  %justification%
               BEGIN
               param _ ELEM #ent3#[cwtype];  %ent3 is a command
               word%
               CASE param OF
                  = 83:  %left%
                     pvalue _ 1;
                  = 84:  %right%
                     pvalue _ 2;
                  = 85:  %center%
                     pvalue _ 3;
```

```
            ENDCASE err(notyet);
         gchngt(jstfct, pvalue, scd, TRUE, FALSE);
         END;
      =2:  %character size%
         BEGIN
         &text _ ELEM #ent3#[tppair];  %ent3 is text%
         gchngt(chsize, &text, scd, TRUE, FALSE);
         END;
      =25:  %font%
         BEGIN
         param _ ELEM #ent3#[cwtype];  %ent3 is a command
         word%
         gchngt(font, param, scd, TRUE, FALSE);
         END;
      =29:  %emphasis%
         BEGIN
         param _ ELEM #ent3#[cwtype];  %ent3 is a command
         word%
         gchngt(29, param, scd, TRUE, FALSE);
         END;
      =26,  %line type%
      =27:  %line style%
         BEGIN
         CASE entty2 OF
            = 100:  %FIGURE%
               fig _ figure;
            = 101:  %RECTANGLE%
               fig _ rectangle;
            = 102:  %LINE%
               fig _ line;
            = 104:  %POINT%
               fig _ point;
            = 105:  %CIRCLE%
               fig _ circle;
            = 106:  %DIAMOND%
               fig _ diamond;
            = 107:  %ITRIANGLE%
               fig _ itriangle;
            = 108:  %ARROWHEAD%
               fig _ arrowhead;
            ENDCASE err(notyet);
         param _ ELEM #ent3#[cwtype];  %ent3 is a command
         word%
         CASE entity OF
            =26:  %line type%
               BEGIN
               CASE param OF
                  = 1:  %solid%
                     pvalue _ vcsolid;
                  = 2:  %dot%
                     pvalue _ vcdot;
                  = 3:  %dot-dash%
                     pvalue _ vcddash;
                  = 4:  %short-dash%
                     pvalue _ vcsdash;
                  = 5:  %long-dash%
```

```
                        pvalue _ vcldash;
                     ENDCASE err(notyet);
                  gchngi(ltype, fig, pvalue, scd, TRUE,
                  FALSE);
                  END;
               =27:  %line style%
                  BEGIN
                  CASE param OF
                     = 1:  %extra fine%
                        pvalue _ lsextra;
                     = 2:  %fine%
                        pvalue _ lsfine;
                     =3:  %regular%
                        pvalue _ lsregular;
                     = 4:  %bold%
                        pvalue _ lsbold;
                     = 5:  %very bold%
                        pvalue _ lsvbold;
                     ENDCASE err(notyet);
                  gchngi(lstyle, fig, pvalue, scd, TRUE,
                  FALSE);
                  END;
               ENDCASE;
            END;
         ENDCASE err(notyet);
      RETURN;
      END.
   (xgcopy) PROCEDURE (ent REF LIST,bug1 REF LIST,bug2 REF
   LIST,frombug REF LIST,tobug REF LIST);                    6C1D3
      %copy%
      LOCAL entity, atpt1, atpt2, from, to;
      %---------------%
      % Convert from FE argument format %
         atpt1 _ gcnvbug ( &bug1);
         atpt2 _ gcnvbug ( &bug2);
         from _ gcnvbug ( &frombug);
         to _ gcnvbug ( &tobug);
         entity _ ELEM #ent#[cwtype];
      CASE entity OF
         = 100:  %FIGURE%
            BEGIN
            gcopy(FALSE, FALSE, figure, atpt1, atpt2, from,
            to);
            END;
         = 27:  %Group of FIGURE%
            BEGIN
            gcopy(FALSE, TRUE, figure, atpt1, atpt2, from,
            to);
            END;
         = 101:  %RECTANGLE%
            BEGIN
            gcopy(FALSE, FALSE, rectangle, atpt1, atpt2, from,
            to);
            END;
         = 102:  %LINE%
            BEGIN
```

```
                gcopy(FALSE, FALSE, line, atpt1, atpt2, from, to);
                END;
            = 103:  %UNCONSTRAINED LINE%
                BEGIN
                gcopy(FALSE, FALSE, line, atpt1, atpt2, from, to);
                END;
            = 104:  %POINT%
                BEGIN
                gcopy(FALSE, FALSE, point, atpt1, atpt2, from,
                to);
                END;
            = 105:  %CIRCLE%
                BEGIN
                gcopy(FALSE, FALSE, circle, atpt1, atpt2, from,
                to);
                END;
            = 106:  %DIAMOND%
                BEGIN
                gcopy(FALSE, FALSE, diamond, atpt1, atpt2, from,
                to);
                END;
            = 107:  %ITRIANGLE%
                BEGIN
                gcopy(FALSE, FALSE, itriangle, atpt1, atpt2, from,
                to);
                END;
            = 108:  %ARROWHEAD%
                BEGIN
                gcopy(FALSE, FALSE, arrowhead, atpt1, atpt2, from,
                to);
                END;
            = cwannotation:  %Statement - that is annotation%
                BEGIN
                gcopy(FALSE, FALSE, annotation, atpt1, atpt2,
                from, to);
                END;
            = 70:  %CELL%
                BEGIN
                err(notyet);
                END;
            = 79:  %DIAGRAM%
                BEGIN
                err(notyet);
                END;
            ENDCASE err(notyet);
        RETURN;
        END.
    (xgcout) PROCEDURE (fentity REF LIST, address REF LIST);
                                                            6C1D4
        LOCAL filename[40], node REF;
        LOCAL STRING locstr[200], errstr[200];
        %----------------------%
        % Convert from FE argument format %
            IF &fentity THEN
                lnkprs( ELEM #fentity#[tppair], $filename) %parse
                link%
```

```
      ELSE filename _ 0;
      &node _ ELEM #address#[tppair];  %address of text
      ptrs%
   % get output file name to locstr %
      IF filename
         THEN % use name supplied by user %
            CASE lnbfls( $filename, 0, $locstr) OF
               = lhostn: NULL;
               ENDCASE
                  err($"Remote File Manipulations Not
                  Implemented Yet")
         ELSE % construct default name %
            err($"Specify name");
   % get a comjfn %
      IF NOT comjfn_sgtjfn(402000000000B,$locstr,$errstr)
         THEN err($errstr);
   % open the file %
      IF NOT SKIP !openf(comjfn, 070000020000B)
         THEN
            BEGIN
            reljfn(comjfn);
            err($"Cannot open file");
            END;
   % setup a gda for the diagram trace out %
   % output the diagram %
      gcout(&node, comxlfb, comxrfb, comytfb, comybfb, 0,
      0, 0, 0);
   % close the file %
      IF NOT SKIP !closf(comjfn)
         THEN err($"Cannot close file");
   % release comjfn %
      reljfn(comjfn);
   RETURN;
   END.

(xgdraw) PROCEDURE (ent REF LIST,ent2 REF LIST,bug1 REF
LIST,bug2 REF LIST,bug3 REF LIST);                    6C1D5
   %draw things%
   LOCAL entity, entty2, pcd1, pcd2, pcd3, jl, ju, il, ir,
   l, w, gi, gda, direct, stid, dsp, scd1, scd2, scd3;
   REF gi,gda;
   %---------------%
   % Convert from FE argument format %
      scd1 _ gcnvbug ( &bug1 );
      scd3 _ gcnvbug ( &bug3 );
      scd2 _ gcnvbug ( &bug2 );
      entity _ ELEM #ent#[cwtype];
      entty2 _ ELEM #ent2#[cwtype];
   &gda _ glda();
   gckemp(&gda);
   gckopn(&gda);  %be sure that some cell is available%
   CASE entity OF
      = 101:  %RECTANGLE%
         BEGIN
         pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
         pcd2 _ gridxy(&gda, scd2.xc, scd2.yc);
```

```
                ju _ maxse((pcd1.yc), (pcd2.yc));
                il _ minse((pcd1.xc), (pcd2.xc));
                w _ absdif(pcd1.xc, pcd2.xc);
                l _ absdif(pcd1.yc, pcd2.yc);
                &gi _ gbldgi(&gda, rectangle, 0, gpack(il, ju), w,
                l : stid, dsp);
                END;
             = 110:  %SQUARE%
                BEGIN
                pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
                pcd2 _ gridxy(&gda, scd2.xc, scd2.yc);
                ju _ maxse((pcd1.yc), (pcd2.yc));
                il _ minse((pcd1.xc), (pcd2.xc));
                w _ MAX (absdif(pcd1.xc, pcd2.xc), absdif(pcd1.yc,
                pcd2.yc));
                &gi _ gbldgi(&gda, rectangle, 0, gpack(il, ju), w,
                w : stid, dsp);
                END;
             = 102:  %LINE%
                BEGIN
                w _ iabs(scd1.xc-scd2.xc);
                l _ iabs(scd1.yc-scd2.yc);
                IF w > l THEN
                   &gi _ gbldgi(&gda, line, 0, gridxy(&gda,
                   scd1.xc, scd1.yc), gda.gdalcd _ gridxy(&gda,
                   scd2.xc, scd1.yc), 0 : stid, dsp)
                ELSE
                   &gi _ gbldgi(&gda, line, 0, gridxy(&gda,
                   scd1.xc, scd1.yc), gda.gdalcd _ gridxy(&gda,
                   scd1.xc, scd2.yc), 0 : stid, dsp);
                END;
             = 19:  %CONTINUATION LINE%
                BEGIN
                pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
                IF absdif(gda.gdalcd.xc, pcd1.xc) >
                absdif(gda.gdalcd.yc, pcd1.yc) THEN
                   BEGIN pcd1.yc _ gda.gdalcd.yc; END
                ELSE
                   BEGIN pcd1.xc _ gda.gdalcd.xc; END;
                &gi _ gbldgi(&gda, line, 0, gda.gdalcd, pcd1, 0 :
                stid, dsp);
                gda.gdalcd _ pcd1;
                END;
             = 87:  %HORIZONTAL LINE%
                BEGIN
                &gi _ gbldgi(&gda, line, 0, gridxy(&gda, scd1.xc,
                scd1.yc), gda.gdalcd _ gridxy(&gda, scd2.xc,
                scd1.yc), 0 : stid, dsp)
                END;
             = 88:  %VERTICAL LINE%
                BEGIN
                &gi _ gbldgi(&gda, line, 0, gridxy(&gda, scd1.xc,
                scd1.yc), gda.gdalcd _ gridxy(&gda, scd1.xc,
                scd2.yc), 0 : stid, dsp);
                END;
             = 103:  %UNCONSTRAINED LINE%
```

```
                BEGIN
                &gi _ gbldgi(&gda, line, 0, gridxy(&gda, scd1.xc,
                scd1.yc), gda.gdalcd _ gridxy(&gda, scd2.xc,
                scd2.yc), 0 : stid, dsp);
                END;
             = 104:  %POINT%
                BEGIN
                &gi _ gbldgi(&gda, point, 0, gridxy(&gda, scd1.xc,
                scd1.yc), 0, 0 : stid, dsp);
                END;
             = 105:  %CIRCLE%
                BEGIN
                pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
                pcd2 _ gridxy(&gda, scd2.xc, scd2.yc);
                CASE entty2 OF
                   = 87:  %HORIZONTALLY TANGENT%
                      BEGIN
                      w _ absdif(pcd1.yc, pcd2.yc);
                      END;
                   = 88:  %VERTICALLY TANGENT%
                      BEGIN
                      w _ absdif(pcd1.xc, pcd2.xc);
                      END;
                   = 89:  %TOUCHING%
                      BEGIN
                      w _ absdif(pcd2.xc, pcd1.xc);
                      l _ absdif(pcd2.yc, pcd1.yc);
                      w _ fix(fsqrt(float((w*w) + (l*l))));
                      END;
                   ENDCASE err(notyet);
                &gi _ gbldgi(&gda, circle, 0, pcd1, w, 0 : stid,
                dsp);
                END;
             = 106:  %DIAMOND%
                BEGIN
                pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
                pcd2 _ gridxy(&gda, scd2.xc, scd2.yc);
                pcd3 _ gridxy(&gda, scd3.xc, scd3.yc);
                ju _ maxse(pcd1.yc, pcd2.yc);
                l _ absdif(pcd1.yc, pcd2.yc) / 2;
                w _ absdif(pcd1.xc, pcd3.xc);
                &gi _ gbldgi(&gda, diamond, 0, gpack(pcd1.xc-w,
                ju-l), w, l : stid, dsp);
                END;
             = 107:  %ITRIANGLE%
                BEGIN
                pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
                pcd2 _ gridxy(&gda, scd2.xc, scd2.yc);
                CASE entty2 OF
                   = 54: % ^ %
                      BEGIN
                      w _ absdif(pcd1.xc, pcd2.xc);
                      l _ absdif(pcd1.yc, pcd2.yc);
                      direct _ north;
                      END;
                   = 55: % v %
```

```
                        BEGIN
                        w _ absdif(pcd1.xc, pcd2.xc);
                        l _ absdif(pcd1.yc, pcd2.yc);
                        direct _ south;
                        END;
                     = 83: % _ %
                        BEGIN
                        l _ absdif(pcd1.xc, pcd2.xc);
                        w _ absdif(pcd1.yc, pcd2.yc);
                        direct _ west;
                        END;
                     = 84: % > %
                        BEGIN
                        l _ absdif(pcd1.xc, pcd2.xc);
                        w _ absdif(pcd1.yc, pcd2.yc);
                        direct _ east;
                        END;
                     ENDCASE err(notyet);
                  &gi _ gbldgi(&gda, itriangle, direct, pcd1, w, l :
                  stid, dsp);
                  END;
               = 108:  %ARROWHEAD%
                  BEGIN
                  CASE entty2 OF
                     = 54: % ^ %
                        direct _ north;
                     = 55: % v %
                        direct _ south;
                     = 83: % _ %
                        direct _ west;
                     = 84: % > %
                        direct _ east;
                     ENDCASE err(notyet);
                  &gi _ gbldgi(&gda, arrowhead, direct, gridxy(&gda,
                  scd1.xc, scd1.yc), gda.gdaawdt, gda.gdaahgt :
                  stid, dsp);
                  END;
               ENDCASE err(notyet);
            IF &gi THEN
               BEGIN
               gpsopn();
               gmvgi(&gda, &gi, gda.gorg, stid, dsp, FALSE);
               gpscls();
               END;
            RETURN;
            END.
         (xgerase) PROCEDURE (ent REF LIST,bug1 REF LIST, bug2 REF
         LIST);                                                    6C1D6
            %delete figures, cells, and diagrams%
            LOCAL entity, atpt1, atpt2, gda REF;
            %---------------%
            % Convert from FE argument format %
               atpt1 _ gcnvbug( &bug1 );
               atpt2 _ gcnvbug( &bug2 );
               entity _ ELEM #ent#[cwtype];
            CASE entity OF
```

```
            = 100:  %FIGURE%
               BEGIN
               gerase(FALSE, figure, atpt1, atpt2);
               END;
            = 27:  %Group of FIGURE%
               BEGIN
               gerase(TRUE, figure, atpt1, atpt2);
               END;
            = 101:  %RECTANGLE%
               BEGIN
               gerase(FALSE, rectangle, atpt1, atpt2);
               END;
            = 102:  %LINE%
               BEGIN
               gerase(FALSE, line, atpt1, atpt2);
               END;
            = 103:  %UNCONSTRAINED LINE%
               BEGIN
               gerase(FALSE, line, atpt1, atpt2);
               END;
            = 104:  %POINT%
               BEGIN
               gerase(FALSE, point, atpt1, atpt2);
               END;
            = 105:  %CIRCLE%
               BEGIN
               gerase(FALSE, circle, atpt1, atpt2);
               END;
            = 106:  %DIAMOND%
               BEGIN
               gerase(FALSE, diamond, atpt1, atpt2);
               END;
            = 107:  %ITRIANGLE%
               BEGIN
               gerase(FALSE, itriangle, atpt1, atpt2);
               END;
            = 108:  %ARROWHEAD%
               BEGIN
               gerase(FALSE, arrowhead, atpt1, atpt2);
               END;
            = cwannotation:  %Statement - that is annotation%
               BEGIN
               gerase(FALSE, annotation, atpt1, atpt2);
               END;
            = 70:  %CELL%
               BEGIN
               err(notyet);
               END;
            = 79:  %DIAGRAM%
               BEGIN
               err(notyet);
               END;
            ENDCASE err(notyet);
         RETURN;
         END.
      (xginit) PROCEDURE ;                                    6C1D7
```

```
    %Initialize for initial entry into graphics subsystem.%
    LOCAL i;
    REF gcommands, garglist, gshwst, gdaphm, gdabas, gbiunt,
    gunitb;
    %---------------%
    % Allocate lists and strings for FE calls %
       IF NOT &gcommands _ gallst(maxfei) THEN
          err($"Cannot allocate commands list - xginit");
       IF NOT &garglist _ gallst(garglsize) THEN
          err($"Cannot allocate argument list - xginit");
       IF NOT (&gshwst _ galstr(2000)) THEN
          err($"Cannot allocate SHOW string - xginit");
    gmakewindow();  %create graphics window%
    gclr();  %clear graphics screen%
    IF logof THEN goutsn();
    IF NOT &gbiunt_galblk(guntsz*guntel) THEN
       err($"System error - cannot get unit table");
    IF NOT &gunitb_galblk(guntsz*guntel) THEN
       BEGIN
       gfrblk(&gbiunt);
       err($"System error - cannot get unit table");
       END;
    glodut(&gbiunt);
    gcoput(&gunitb, &gbiunt);
    gbuabb _ &gunitb;  %default basic unit%
    IF NOT &gdaphm_galblk(gdalgt) THEN
       err($"System error - cannot get gda");
    IF NOT &gdabas_galblk(gdalgt) THEN
       BEGIN
       gfrblk(&gdaphm);
       err($"System error - cannot get gda");
       END;
    glodda(&gdaphm);
    gcopda(&gdabas, &gdaphm);
    galoat(gatsiz);
    RETURN;
    END.
(xginpt) PROCEDURE (fentity REF LIST, addrs1 REF LIST,
addrs2 REF LIST);                                          6C1D8
    %Input a diagram in SPL format from the specified block
    of the sequential file listed and store it as an NLS
    diagram connected to the statement provided.%
    LOCAL filename[40], tpblock REF, tp2 REF, tpstat REF,
    gda REF, n, msgerr;
    LOCAL STRING locstr[200], errstr[200];
    %---------------%
    % Convert from FE argument format %
       IF &fentity THEN
          lnkprs( ELEM #fentity#[tppair], $filename); %parse
          link%
       IF &addrs1 THEN
          &tpblock _ ELEM #addrs1#[tppair]
       ELSE &tpblock _ 0;
       &tpstat _ ELEM #addrs2#[tppair];
    % get input file name to locstr %
       IF &fentity
```

```
            THEN % use name supplied by user %
               CASE lnbfls( $filename, 0, $locstr) OF
                  = lhostn: NULL;
                  ENDCASE
                     err($"Remote File Manipulations Not
                     Implemented Yet")
            ELSE % construct default name %
               err($"Specify name");
         % get a spljfn %
          IF NOT spljfn_sgtjfn(0B,$locstr,$errstr)
            THEN err($errstr);
         % open the file %
          IF NOT SKIP !openf(spljfn, 070000200000B)
            THEN
               BEGIN
               reljfn(spljfn);
               spljfn_0;
               err($"Cannot open file");
               END;
         % get the block number %
          IF &tpblock THEN
            BEGIN
            &tp2 _ &tpblock + d2sel;
            *locstr* _ tpblock tp2;
            n _ VALUE($locstr);
            IF n <= 0 THEN n _ 1;
            END
          ELSE
            n _ 1;
         % check to see if the block is on the file %
          IF NOT splfb(n) THEN
            err($"SPL block not found");
         % create the diagram %
          &gda _ glda();
          gdgman(&gda, gcreate, tpstat, 0);
         % read the SPL data %
          msgerr _ splrd(&gda);
         % mark the statement and recreate the display %
          grecred();
          dpset(dsprfmt, tpstat, endfil, endfil);
          ST tpstat _ SF(tpstat) SE(tpstat),  " (see attached
          diagram)";
         % note any errors on the statement %
          IF msgerr THEN
            BEGIN
            ST tpstat _ SF(tpstat) SE(tpstat), EOL,
               "ERRORS in BLOCK ", STRING(n), EOL,
               "(one or more of the following encountered)";
            IF msgerr .A sple01 THEN
               ST tpstat _ SF(tpstat) SE(tpstat), EOL,
                  "-margin error";
            IF msgerr .A sple02 THEN
               ST tpstat _ SF(tpstat) SE(tpstat), EOL,
                  "-unknown command";
            IF msgerr .A sple03 THEN
               ST tpstat _ SF(tpstat) SE(tpstat), EOL,
```

```
                    "-line error";
                IF msgerr .A sple04 THEN
                   ST tpstat _ SF(tpstat) SE(tpstat), EOL,
                      "-text error";
                IF msgerr .A sple05 THEN
                   ST tpstat _ SF(tpstat) SE(tpstat), EOL,
                      "-keyword unknown";
                IF msgerr .A sple06 THEN
                   ST tpstat _ SF(tpstat) SE(tpstat), EOL,
                      "-keyword value error";
                IF msgerr .A sple07 THEN
                   ST tpstat _ SF(tpstat) SE(tpstat), EOL,
                      "-data range error";
                END;
             %  This code was executed in cleanup mode in NLS-8.5 -
             it is commented out
                IF spljfn THEN
                   BEGIN
                   %% close the file %%
                      IF NOT SKIP !closf(spljfn)
                         THEN err($"Cannot close file");
                   %% release spljfn %%
                      reljfn(spljfn);
                      spljfn_0;
                   END;
             %
             RETURN;
             END.
          (xglabl) PROCEDURE (bug1 REF LIST,address REF LIST);   6C1D9
             %caption diagrams - lable mode%
             LOCAL scd1, text REF, stid, gda, i1, pcd1, mchars,
             nlines, depth, width;
             LOCAL TEXT POINTER tp1, tp2;
             REF gda;
             %---------------%
             scd1 _ gcnvbug( &bug1 );
             &gda _ gdacds(scd1);
             gckemp(&gda);
             gckopn(&gda);
             &text _ ELEM #address#[tppair];  %addr. of text pointer
             block%
             tp1 _ text;
             tp1[1] _ text[1];
             tp2 _ text [2];
             tp2[1] _ text[3];
             mchars _ gtfit($tp1, $tp2 : nlines);
             pcd1 _ gridxy(&gda, scd1.xc, scd1.yc);
             width _ (2*gda.gdabdr) + (mchars*gda.gdacw);
             depth _ (2*gda.gdabdr) + (nlines*gda.gdach);
             i1.yc _ gsigne(pcd1.yc) + gda.gdach + gda.gdabdr;
             CASE gda.gdajus OF
                =1: %left%
                   BEGIN
                   i1.xc _ gsigne(pcd1.xc) - gda.gdabdr;
                   END;
                =2: %right%
```

```
            BEGIN
            i1.xc _ gsigne(pcd1.xc) + gda.gdabdr - width;
            END;
         =3: %center%
            BEGIN
            i1.xc _ gsigne(pcd1.xc) - width/2;
            END;
         ENDCASE;
      stid _ gbldtn(&gda, i1, width, depth, $tp1, $tp2);
      gpsopn();
      gmvtn(&gda, stid, gda.gorg, FALSE);
      gpscls();
      RETURN;
      END.
   (xglong) PROCEDURE ( ent REF LIST, bug1 REF LIST, tobug REF
   LIST);                                                    6C1D10
      %Handle the extension and contraction (backspace) of
      long lines.%
      LOCAL atpt1, to, entity, gda, gi, atadr, stid, dsp;
      REF gda, gi, atadr;
      %---------------%
      % Convert from FE argument format %
         entity _ ELEM #ent#[cwtype];
         atpt1 _ gcnvbug( &bug1 );
         to _ gcnvbug( &tobug );
      CASE entity OF
         =20, =22: %EXTEND Visibly or Invisibly%
            BEGIN
            IF NOT &atadr _ gfind(line, atpt1 : &gda, stid,
            dsp) THEN
               err($"line not found");
            gckemp(&gda);
            gckopn(&gda);
            gpsopn();
            gmvdel(&atadr, FALSE);
            &gi _ gexlgi(stid, dsp, entity, gda.gdalcd _
            gridxy(&gda, to.xc, to.yc), gda.glstid : stid,
            dsp);
            IF &gi THEN
               gmvgi(&gda, &gi, gda.gorg, stid, dsp, FALSE);
            gpscls();
            END;
         = 21:  %BACKSPACE%
            BEGIN
            IF NOT &atadr _ gfind(line, atpt1 : &gda, stid,
            dsp) THEN
               err($"line not found");
            gckemp(&gda);
            gckopn(&gda);
            gpsopn();
            gmvdel(&atadr, FALSE);
            &gi _ gexlgi(stid, dsp, entity, 0, gda.glstid :
            stid, dsp);
            IF &gi THEN
               gmvgi(&gda, &gi, gda.gorg, stid, dsp, FALSE);
            gpscls();
```

```
            END;
         ENDCASE err(notyet);
      RETURN;
      END.
   (xgmove) PROCEDURE (ent REF LIST,bug1 REF LIST, bug2 REF
   LIST, frombug REF LIST, tobug REF LIST);                    6C1D11
      %Move%
      LOCAL entity,atpt1, atpt2, from, to, gda REF;
      %---------------%
      % Convert from FE argument format %
         entity _ ELEM #ent#[cwtype];
         atpt1 _ gcnvbug( &bug1 );
         atpt2 _ gcnvbug( &bug2 );
         from _ gcnvbug( &frombug );
         to _ gcnvbug( &tobug );
      CASE entity OF
         = 100:  %FIGURE%
            BEGIN
            gmove(FALSE, figure, atpt1, atpt2, from, to);
            END;
         = 27:  %Group of FIGURE%
            BEGIN
            gmove(TRUE, figure, atpt1, atpt2, from, to);
            END;
         = 101:  %RECTANGLE%
            BEGIN
            gmove(FALSE, rectangle, atpt1, atpt2, from, to);
            END;
         = 102:  %LINE%
            BEGIN
            gmove(FALSE, line, atpt1, atpt2, from, to);
            END;
         = 103:  %UNCONSTRAINED LINE%
            BEGIN
            gmove(FALSE, line, atpt1, atpt2, from, to);
            END;
         = 104:  %POINT%
            BEGIN
            gmove(FALSE, point, atpt1, atpt2, from, to);
            END;
         = 105:  %CIRCLE%
            BEGIN
            gmove(FALSE, circle, atpt1, atpt2, from, to);
            END;
         = 106:  %DIAMOND%
            BEGIN
            gmove(FALSE, diamond, atpt1, atpt2, from, to);
            END;
         = 107:  %ITRIANGLE%
            BEGIN
            gmove(FALSE, itriangle, atpt1, atpt2, from, to);
            END;
         = 108:  %ARROWHEAD%
            BEGIN
            gmove(FALSE, arrowhead, atpt1, atpt2, from, to);
            END;
```

```
            = cwannotation:  %Statement - that is annotation%
               BEGIN
               gmove(FALSE, annotation, atpt1, atpt2, from, to);
               END;
            = 70:  %CELL%
               BEGIN
               err(notyet);
               END;
            = 79:  %DIAGRAM%
               BEGIN
               err(notyet);
               END;
            ENDCASE err(notyet);
         RETURN;
         END.
      (xgorg) PROCEDURE (ent REF LIST, addrs1 REF LIST, addrs2
      REF LIST);                                                    6C1D12
         %Initiate diagrams, cells, and subcells%
         LOCAL entity, tp1 REF, tp2 REF, gda REF, clear;
         %---------------%
         % Convert from FE argument format %
            entity _ ELEM #ent#[cwtype];
            &tp1 _ ELEM #addrs1# [tppair];
            IF &addrs2 THEN
               &tp2 _ ELEM #addrs2# [tppair];
         CASE entity OF
            = 79:  %DIAGRAM%
               BEGIN
               &gda _ glda();
               clear _ gda.gdstid;  %save to see if screen must
               be cleared%
               gdgman(&gda, gcreate, tp1, 0);
               IF clear THEN
                  grecred()  %previous diagram protrayed in
                  viewport%
               ELSE
                  BEGIN  %viewport had been empty%
                     gsetpv(&gda);
                     goutmgn(&gda);
                     gdspdg(&gda);
                  END;
               dpset(dsprfmt, tp1, endfil, endfil);
               ST tp1 _ SF(tp1) SE(tp1),  " (see attached
               diagram)";
               END;
            ENDCASE err(notyet);
         RETURN;
         END.
      (xgport) PROCEDURE (address REF LIST, bug1 REF LIST); 6C1D13
         %Portray the diagram at tp1 in the viewport at scd.%
         LOCAL tp1 REF, scd, clear,gda REF, blkadr REF, stdb;
         %---------------%
         % Convert from FE argument format %
            scd _ gcnvbug( &bug1 );
            &tp1 _ ELEM #address#[tppair];
         &gda _ glda();  %not correct fix%
```

```
      clear _ gda.gdstid;  %save to see if screen must be
      cleared%
      IF NOT lodprop(tp1, dhtyp : &blkadr, stdb) THEN
      err($"diagram not found");
      gda.gdstid _ tp1;
      gda.gtstid _ getitree(stdb);
      gda.gcstid _ 0;
      gda.glstid _ 0;
      gda.gorg _ 0;
      gda.gmleft   _ blkadr[sdbhdl].gmargleft;
      gda.gmright  _ blkadr[sdbhdl].gmargright;
      gda.gmtop    _ blkadr[sdbhdl].gmargtop;
      gda.gmbottom _ blkadr[sdbhdl].gmargbottom;
      gda.gwleft   _ float(gsigne(gda.gmleft));
      gda.gwright  _ float(gsigne(gda.gmright));
      gda.gwtop    _ float(gsigne(gda.gmtop));
      gda.gwbottom _ float(gsigne(gda.gmbottom));
      %NOTE:  CODE FOR OLD FILES.  If old file (length of
      diagram header block is less than current length), use
      default resolution.  Otherwise use resolution from
      property block%
         gda.gdares _ IF getdblen(stdb) - sdbhdl < gdhsiz
            THEN dfltr
            ELSE blkadr[sdbhdl].gdhres;
      IF clear THEN
         grecred()  %previous diagram protrayed in viewport%
      ELSE
         BEGIN  %viewport had been empty%
            gsetpv(&gda);
            goutmgn(&gda);
            gdspdg(&gda);
         END;
      RETURN;
      END.
   (xgref) PROCEDURE (tp1);                                  6C1D14
      %Refer to a cell%
      REF result,tp1;
      %---------------%
      err(notyet);
      RETURN;
      END.
   (xgrentr) PROCEDURE ;                                     6C1D15
      %Initialize for re-entry into graphics subsystem.%
      LOCAL i;
      REF result;
      %---------------%
      NULL;
      RETURN;
      END.
   (xgreset) PROCEDURE (ent REF LIST);                       6C1D16
      %Reset to user or system default%
      LOCAL entity, gda REF, propdata[3];
      %---------------%
      % convert from fe argument format %
         entity _ ELEM #ent#[cwtype];
      CASE entity OF
```

```
        =18:  %drawspecs%
           BEGIN
           &gda _ glda();
           gsdflt(&gda);
           END;
        ENDCASE err(notyet);
     RETURN ;
     END.
  (xgset) PROCEDURE (ent REF LIST, ent2 REF LIST, ent3 REF
  LIST);                                                    6C1D17
     %setup various environment variables%
     LOCAL tp1 REF, tp2 REF, entity, param, scd, gda REF,
     pvalue, cds;
     %---------------%
     % Convert from FE argument format %
        entity _ ELEM #ent#[cwtype];
     CASE entity OF
        =10:  %justification type%
           BEGIN
           param _ ELEM #ent2#[cwtype];  %ent2 is a command
           word%
           &gda _ glda();
           CASE param OF
              = 83:  %left%
                 gda.gdajus _ 1;
              = 84:  %right%
                 gda.gdajus _ 2;
              = 85:  %center%
                 gda.gdajus _ 3;
              ENDCASE;
           END;
        =29:  %emphasis byte%
           BEGIN
           &gda _ glda();
           param _ ELEM #ent2#[cwtype];  %ent2 is a command
           word%
           CASE param OF
              = 1:  %medium%
                 gda.gdaemp.gempb _ 0;
              = 3:  %light%
                 gda.gdaemp.gempb _ 1;
              = 4:  %bold%
                 gda.gdaemp.gempb _ 2;
              = 5:  %slanted%
                 gda.gdaemp.gemps _ 1;
              = 6:  %straight%
                 gda.gdaemp.gemps _ 0;
              = 7:  %underlined%
                 gda.gdaemp.gempu _ 1;
              = 8:  %not underlined%
                 gda.gdaemp.gempu _ 0;
              = 9:  %monospaced%
                 gda.gdaemp.gempm _ 1;
              = 10:  %proportionally spaced%
                 gda.gdaemp.gempm _ 0;
              ENDCASE;
```

```
      END;
   =16:  %margin%
      BEGIN
      &gda _ glda();
      gckemp(&gda);
      param _ ELEM #ent2#[cwtype];  %ent2 is a command
      word%
      IF &ent3 THEN
         BEGIN
         scd _ gcnvbug( &ent3 );  %ent3 is a point%
         cds _ gridxy(&gda, scd.xc, scd.yc);
         END;
      gmgnfl _ TRUE;
      CASE param OF
         = 62:  %on%
            gmgnfl _ TRUE; %margins on%
         = 63:  %off%
            gmgnfl _ FALSE; %margins off%
         = 83:  %left%
            IF gsigne(cds.xc)<gsigne(gda.gmright)
               THEN gda.gmleft _ cds.xc;
         = 84:  %right%
            IF gsigne(cds.xc)>gsigne(gda.gmleft)
               THEN gda.gmright _ cds.xc;
         = 94:  %top%
            IF gsigne(cds.yc)>gsigne(gda.gmbottom)
               THEN gda.gmtop _ cds.yc;
         = 95:  %bottom%
            IF gsigne(cds.yc)<gsigne(gda.gmtop)
               THEN gda.gmbottom _ cds.yc;
      ENDCASE err(notyet);
      grpdhd(&gda);  %reprop diagram header block%
      goutmgn(&gda);
      END;
   =17:  %grid size%
      BEGIN
      &gda _ glda();
      &tp1 _ ELEM #ent2# [tppair];  %ent2 is text%
      gda.gdagrd _ gtxtpg(&tp1, &gda);
      END;
   =108:  %arrowhead size%
      BEGIN
      &gda _ glda();
      &tp1 _ ELEM #ent2# [tppair];  %ent2 is text%
      &tp2 _ ELEM #ent3# [tppair];  %ent3 is text%
      %store heigth and half width%
         gda.gdaahgt _ gsetas(&gda, &tp1, &tp2:
         gda.gdaawdt);
      END;
   =2:  %character size%
      BEGIN
      &gda _ glda();
      &tp1 _ ELEM #ent2# [tppair];  %ent2 is text%
      gda.gdach _ gsetcs(&gda, &tp1: gda.gdacw);
      END;
   =25:  %font%
```

```
            BEGIN
            &gda _ glda();
            param _ ELEM #ent2#[cwtype];  %ent2 is a command
            word%
            gda.gdafnt _ gsetf(param);
            END;
         =122:  %viewgraph mode%
            BEGIN
            param _ ELEM #ent2#[cwtype];  %ent2 is a command
            word%
            IF param=63 %off% THEN gblrrc_1 ELSE gblrrc_7;
            END;
         =26:  %line type%
            BEGIN
            param _ ELEM #ent2#[cwtype];  %ent2 is a command
            word%
            &gda _ glda();
            CASE param OF
               = 1:  %solid%
                  gda.gdalvec _ vcsolid;
               = 2:  %dot%
                  gda.gdalvec _ vcdot;
               = 3:  %dot-dash%
                  gda.gdalvec _ vcddash;
               = 4:  %short-dash%
                  gda.gdalvec _ vcsdash;
               = 5:  %long-dash%
                  gda.gdalvec _ vcldash;
               ENDCASE err(notyet);
            END;
         =27:  %line style%
            BEGIN
            param _ ELEM #ent2#[cwtype];  %ent2 is a command
            word%
            &gda _ glda();
            CASE param OF
               = 1:  %extra fine%
                  pvalue _ lsextra;
               = 2:  %fine%
                  pvalue _ lsfine;
               =3:  %regular%
                  pvalue _ lsregular;
               = 4:  %bold%
                  pvalue _ lsbold;
               = 5:  %very bold%
                  pvalue _ lsvbold;
               ENDCASE err(notyet);
            gda.gdalint _ lstabl[(pvalue - 1)*2];
            gda.gdalbrd _ lstabl[(pvalue - 1)*2 + 1];
            END;
         =28:  %terminal type%
            BEGIN
            param _ ELEM #ent2#[cwtype];  %ent2 is a command
            word%
            CASE param OF
               = 1:  %t4014%
```

```
                    gdevtyp _ t4014;
                 = 2:  %t4012%
                    gdevtyp _ t4012;
                 ENDCASE err(notyet);
              END;
           ENDCASE err(notyet);
        RETURN;
        END.
     (xgshow) PROCEDURE (ent REF LIST,ent2 REF LIST, ent3 REF
     LIST);                                                    6C1D18
        %Show things.%
        LOCAL scd, scd2, entity, param, gda REF, fig;
        %---------------%
        % convert from fe argument format %
           entity _ ELEM #ent#[cwtype];
        CASE entity OF
           =17:  %grid%
              BEGIN
              scd _ gcnvbug( &ent2 );  %ent2 - point selections%
              scd2 _ gcnvbug( &ent3 );  %ent3 - point
              selections%
              &gda _ gdacds(scd2);
              gckemp(&gda);
              gsgrid(&gda, scd, scd2);
              END;
           =18:  %drawspecs%
              BEGIN
              scd _ gcnvbug( &ent2 );  %ent2 - point selections%
              &gda _ glda();
              gsdrws(&gda);
              END;
           =30:  %specific drawspecs%
              BEGIN
              param _ ELEM #ent2#[cwtype];  %ent2 is a command
              word%
              scd _ gcnvbug( &ent3 );  %ent3  is a point
              selection%
              IF param = cwannotation %statement%
                 THEN gchngt(0, 0, scd, FALSE, TRUE)
                 ELSE
                    BEGIN
                    CASE param OF
                       = 100:  %FIGURE%
                          fig _ figure;
                       = 101:  %RECTANGLE%
                          fig _ rectangle;
                       = 102:  %LINE%
                          fig _ line;
                       = 104:  %POINT%
                          fig _ point;
                       = 105:  %CIRCLE%
                          fig _ circle;
                       = 106:  %DIAMOND%
                          fig _ diamond;
                       = 107:  %ITRIANGLE%
                          fig _ itriangle;
```

```
                        = 108:  %ARROWHEAD%
                            fig _ arrowhead;
                        ENDCASE err(notyet);
                    gchngi(0, fig, 0, scd, FALSE, TRUE);
                    END;
                END;
            =24:  %units%
                BEGIN
                &gda _ glda();
                gsunit(&gda);
                END;
            ENDCASE err(notyet);
        RETURN ;
        END.
    (xgterm) PROCEDURE ;                                        6C1D19
        %Terminate the graphics subsystem.%
        LOCAL i, next;
        REF gcommands, garglist, gshwst, gprwndw, result,
        gdabas, gdaphm, gbiunt, gunitb;
        %---------------%
        dfedew();  %delete FE graphics window%
        gfrewa();  %return wa storage%
        gfreat();
        gfrblk(&gbiunt);  %return unit table storage%
        gfrblk(&gunitb);
        gfrblk(&gdaphm);
        UNTIL NOT &gdabas DO
            BEGIN
            next_gdabas.gdanxt;
            gfrblk(&gdabas);
            &gdabas_next;
            END;
        gfrlst(&gcommands)  %deallocate batch commands commands
        list%;
        gfrlst(&garglist)  %deallocate FE argument list%;
        delwa(&gprwndw);  %delete wa%
        gfrstr(&gshwst);  %deallocate show commands string%
        RETURN ;
        END.
    (xgtest) PROCEDURE;                                         6C1D20
        LOCAL x1,y1,x2,y2;
        LOCAL STRING s[5];
        %---------------%
        *s* _ "Hello";
        gclr();
        gpsopn();
        wtrect(0,0,4095,2932);
        FOR x1_2100 UP 400 UNTIL >3900 DO
            FOR y1_40 UP 400 UNTIL >2800 DO
                wtrect(x1,y1,x1+200,y1+200);
        wtline(0,1466,4095,1466);
        wtline(2047,0,2047,2932);
        wtmark(400,400);
        wtarc(1000,2400,400);
        wtchar(800,800,5,$s);
        gpscls();
```

```
      RETURN ;
      END.
   (xgtran) PROCEDURE ;                                          6C1D21
      %Transform cells%
      REF result;
      %---------------%
      err(notyet);
      RETURN;
      END.
   (xgupda) PROCEDURE ;                                          6C1D22
      %Recreate all graphics da's%
      REF result;
      %---------------%
      grecred();
      RETURN;
      END.
   (xgvprt) PROCEDURE (ent REF LIST,bug1 REF LIST,bug2 REF
   LIST);                                                        6C1D23
      %Viewport management%
      LOCAL gda REF,entity,scd1,scd2,
         edgecd,
         vebug,  %TRUE if vertical edge bugged, else FALSE%
         gdanew REF,
         gdangh REF,  %gda for viewport sharing edge to be
         deleted%
         whiche,  %which edge (left, right, top, bottom)%
         xedge,
         yedge;
      %---------------%
      % convert from fe argument format %
         entity _ ELEM #ent#[cwtype];
         scd1 _ gcnvbug( &bug1 );
         scd2 _ gcnvbug( &bug2 );
      CASE entity OF
         = 15:  %erase, i.e. delete edge%
            BEGIN
            &gda_glda( : edgecd);
            vebug _ gdahv(&gda, scd1: whiche);
            %get a point on the edge to be deleted%
            edgecd.xc _ IF vebug
               THEN
                  IF whiche=ledge
                     THEN gda.gleft
                     ELSE gda.gright
               ELSE scd1.xc;
            edgecd.yc _ IF vebug
               THEN scd1.yc
               ELSE IF whiche=tedge
                  THEN gda.gtop
                  ELSE gda.gbottom;
            %get neighboring viewport%
               &gdangh _ gnghbr(&gda, edgecd);
               IF ( vebug AND (gda.gtop # gdangh.gtop OR
               gda.gbottom # gdangh.gbottom)) OR
               ( NOT vebug AND (gda.gleft # gdangh.gleft OR
               gda.gright # gdangh.gright)) THEN
```

```
                        err($"Cannot delete edge shared by more than
                        2 viewports");
                     %expand viewport to be kept%
                        CASE whiche OF
                           =ledge:  gda.gleft _ gdangh.gleft;
                           =redge:  gda.gright _ gdangh.gright;
                           =tedge:  gda.gtop _ gdangh.gtop;
                           =bedge:  gda.gbottom _ gdangh.gbottom;
                        ENDCASE;
                     gdelda(&gdangh);
                     END;
                  = 96:  %Perpendicular, i.e. insert edge%
                     BEGIN
                     &gda_glda( : edgecd);
                     vebug _ gdahv(&gda, scd1: whiche);
                     IF gdacds(scd1) # &gda THEN
                        err($"Edge not in viewport");
                     IF NOT &gdanew_galblk(gdalgt) THEN
                        err($"Cannot make any more edges");
                     gcopda(&gdanew, &gda);
                     glnkda(&gdanew);
                     gdanew.gdstid _ 0;  %no diagram in new viewport%
                     xedge_ scd1.xc;
                     yedge_ scd1.yc;
                     IF vebug THEN
                        BEGIN  %Splitting viewport horizontally%
                        IF edgecd.yc<yedge THEN
                           BEGIN  %New viewport is in bottom part of
                           old%
                           gda.gtop _ yedge;
                           gdanew.gbottom _ yedge;
                           END
                        ELSE
                           BEGIN  %New viewport is in top part of old%
                           gda.gbottom _ yedge;
                           gdanew.gtop _ yedge;
                           END;
                        END
                     ELSE
                        BEGIN  %Splitting viewport vertically%
                        IF edgecd.xc<xedge THEN
                           BEGIN  %New viewport is in right part of
                           old%
                           gda.gright _ xedge;
                           gdanew.gleft _ xedge;
                           END
                        ELSE
                           BEGIN  %New viewport is in left part of old%
                           gda.gleft _ xedge;
                           gdanew.gright _ xedge;
                           END;
                        END;
                     END;
                  ENDCASE err(notyet);
               grecred();
               RETURN;
```

```
    END.
(xgwdom) PROCEDURE (ent REF LIST,bug1 REF LIST,bug2 REF
LIST);                                                    6C1D24
    %manipulate the window to effect pan and zoom%
    LOCAL entity, scd1, scd2, i, x, y, gda REF, cpair[4],
    blkadr REF, stdb;
    %---------------%
    % convert from fe argument format %
       entity _ ELEM #ent#[cwtype];
       scd1 _ IF &bug1 THEN gcnvbug( &bug1 ) ELSE 0;
       scd2 _ IF &bug2 THEN gcnvbug( &bug2 ) ELSE 0;
    &gda _ glda();
    gckemp(&gda);
    CASE entity OF
       = 90:  %IN%
          BEGIN
          x _ fdiv( fsub(gda.gwright, gda.gwleft), ffour);
          y _ fdiv( fsub(gda.gwtop, gda.gwbottom), ffour);
          gda.gwleft   _ fadd(gda.gwleft, x);
          gda.gwright  _ fsub(gda.gwright, x);
          gda.gwtop    _ fsub(gda.gwtop, y);
          gda.gwbottom _ fadd(gda.gwbottom, y);
          END;
       = 91:  %OUT%
          BEGIN
          x _ fdiv( fsub(gda.gwright, gda.gwleft), ftwo);
          y _ fdiv( fsub(gda.gwtop, gda.gwbottom), ftwo);
          gda.gwleft   _ fsub(gda.gwleft, x);
          gda.gwright  _ fadd(gda.gwright, x);
          gda.gwtop    _ fadd(gda.gwtop, y);
          gda.gwbottom _ fsub(gda.gwbottom, y);
          END;
       = 93:  %FULL%
          BEGIN
          IF NOT lodprop(gda.gdstid, dhtyp : &blkadr, stdb)
          THEN err($"diagram not found");
          gda.gwleft _
          float(gsigne(blkadr[sdbhdl].gmargleft));
          gda.gwright _
          float(gsigne(blkadr[sdbhdl].gmargright));
          gda.gwtop _
          float(gsigne(blkadr[sdbhdl].gmargtop));
          gda.gwbottom _
          float(gsigne(blkadr[sdbhdl].gmargbottom));
          END;
       = 92:  %TO%
          BEGIN
          gtranf(scd1, $cpair   , &gda+gdapx);
          gtranf(scd2, $cpair[2], &gda+gdapx);
          gda.gwleft   _ MIN(cpair   , cpair[2]);
          gda.gwright  _ MAX(cpair   , cpair[2]);
          gda.gwbottom _ MIN(cpair[1], cpair[3]);
          gda.gwtop    _ MAX(cpair[1], cpair[3]);
          END;
       = 14:  %MOVE%
          BEGIN
```

```
                gtranf(scd1, $cpair   , &gda+gdapx);
                gtranf(scd2, $cpair[2], &gda+gdapx);
                x _ fsub(cpair   , cpair[2]);
                y _ fsub(cpair[1], cpair[3]);
                gda.gwleft   _ fadd(gda.gwleft, x);
                gda.gwright  _ fadd(gda.gwright, x);
                gda.gwtop    _ fadd(gda.gwtop, y);
                gda.gwbottom _ fadd(gda.gwbottom, y);
                END;
             = 85:  %CENTER%
                BEGIN
                gtranf(scd1, $cpair   , &gda+gdapx);
                cpair[2] _ fadd(gda.gwleft,fdiv(fsub(gda.gwright,
                gda.gwleft), ftwo));
                cpair[3] _ fadd(gda.gwbottom,fdiv(fsub(gda.gwtop,
                gda.gwbottom), ftwo));
                x _ fsub(cpair   , cpair[2]);
                y _ fsub(cpair[1], cpair[3]);
                gda.gwleft   _ fadd(gda.gwleft, x);
                gda.gwright  _ fadd(gda.gwright, x);
                gda.gwtop    _ fadd(gda.gwtop, y);
                gda.gwbottom _ fadd(gda.gwbottom, y);
                END;
             ENDCASE err(notyet);
          grecred();
          RETURN;
          END.
    %FINISH.%%% % end of x-routine file for split version of
```

```
subsystem%
%Execution routine support%
   (gchngi) PROCEDURE (entity, fig, param, scd, chngfl,
   showfl);                                                    6C1F1
      %Change or show graphics instruction parameters%
      LOCAL gi, workarea, stid, gda, dsp, atadr;
      REF gi, gda, atadr;
      &gda _ glda();
      gckemp(&gda);
      gckopn(&gda);
      IF NOT &atadr _ gfind(fig, scd : &gda, stid, dsp) THEN
         err($"figure not found - gchngi");
      workarea _ glodwa(stid);
      &gi _ gwaut[workarea].gwaaddr + gwahdl+ dsp;
      IF showfl THEN %show current parameters%
         BEGIN
         gfbctl(typenullit);  %clear text screen%
         gdlfmt(gi.gphvec, gi.gphint, gi.gphbrd);  %Line type
         and style%
         gfbctl(addcalit);  %wait for CA or CD and then
         refresh screen%
         END;
      IF chngfl THEN %change current parameters%
         BEGIN
         CASE entity OF
            = ltype:  %line type%
               gi.gphvec _ param;
            = lstyle:  %line style%
               BEGIN
               gi.gphint _ lstabl[(param-1)*2];
               gi.gphbrd _ lstabl[(param-1)*2+1];
               END;
            ENDCASE;
         gputwa(workarea);
         gpsopn();
         gmvdel(&atadr, TRUE);
         gmvgi(&gda, &gi, gda.gorg, stid, dsp, FALSE);
         gpscls();
         END;
      RETURN;
      END.
   (gchngt) PROCEDURE (entity, param, scd, chngfl, showfl);
                                                               6C1F2
      %The text format is changed  or displayed for the
      statement at scd%
      LOCAL atadr, gda, stid, dsp, blkadr, stdb, tfold[5],
      tfnew[5];
      REF param, atadr, gda, blkadr;
      &gda _ glda();
      gckemp(&gda);
      gckopn(&gda);
      IF NOT &atadr _ gfind(annotation, scd : &gda, stid, dsp)
      THEN
         err($"statement not found - gchngt");
      IF NOT lodprop(stid, gtftyp : &blkadr, stdb) THEN
         err($"gtftyp not found - gchngt");
```

```
        mvbfbf (&blkadr+sdbhdl, $tfold, 5);  %save old text
        format%
        mvbfbf (&blkadr+sdbhdl, $tfnew, 5);  %initialize new
        text format%
        IF showfl THEN %show current text format%
           BEGIN
           gfbctl(typenullit);  %clear text screen%
           gdtfmt($gda, tfold.gtfch, tfold.gtfcw, tfold.gtfjus,
           tfold.gtffnt, tfold.gtfemp);
           gfbctl(addcalit);  %wait for CA or CD and then
           refresh screen%
           END;
        IF chngfl THEN %change the text format%
           BEGIN
           CASE entity OF
              =jstfct:  %justification%
                 tfnew.gtfjus _ &param;
              =29:  %emphasis byte%
                 BEGIN
                 CASE param OF
                    = 1:  %medium%
                       tfnew.gtfemp.gempb _ 0;
                    = 3:  %light%
                       tfnew.gtfemp.gempb _ 1;
                    = 4:  %bold%
                       tfnew.gtfemp.gempb _ 2;
                    = 5:  %slanted%
                       tfnew.gtfemp.gemps _ 1;
                    = 6:  %straight%
                       tfnew.gtfemp.gemps _ 0;
                    = 7:  %underlined%
                       tfnew.gtfemp.gempu _ 1;
                    = 8:  %not underlined%
                       tfnew.gtfemp.gempu _ 0;
                    = 9:  %monospaced%
                       tfnew.gtfemp.gempm _ 1;
                    = 10:  %proportionally spaced%
                       tfnew.gtfemp.gempm _ 0;
                    ENDCASE;
                 END;
              =font:  %font%
                 BEGIN
                 tfnew.gtffnt _ gsetf(param);
                 %SHOULD CAUSE OTHER TEXT FORMAT FIELDS TO
                 CHANGE TOO%
                 END;
              =chsize:  %character size%
                 BEGIN
                 tfnew.gtfch _ gsetcs(&gda, &param:
                 tfnew.gtfcw);
                 gntwnd($tfold, $tfnew, stid);
                 END;
              ENDCASE;
           reprop(stid, gtftyp, 5, $tfnew);
           gpsopn();
           gmvdel(&atadr, TRUE);
```

```
      gmvtn(&gda, stid, gclorg(stid), FALSE);
      gpscls();
      END;
   RETURN;
   END.
(gclorg) PROCEDURE (stid);                                    6C1F3
   %Given an stid find the coordinates of the cell
   containing the node.%
   RETURN(0);
   END.
(gcnvbug)   % convert front-end cursor coordinates %
PROCEDURE (bug REF LIST % => crdin %);                        6C1F4
   % Procedure description
      FUNCTION
         Convert front-end point selection to coordinates
         for graphics back-end.  Set global for current
         coordinates (last point bugged on graphics
         screen).
      ARGUMENTS
         bug:  front-end point selection
      RESULTS
         crdin:  converted coords (xcoordinate in LH,
         ycoordinate in RH)
      NON-STANDARD CONTROL
         none
      GLOBALS
         Set
            glastxy
      %
   % Declarations %
      LOCAL crdin;
   % Get coordinates from point selection list and convert
   to internal coordinate system %
      crdin.xc _ glastxy.xc _ (ELEM #bug#[xcoord] .A
      777777B)*4;
      crdin.yc _ glastxy.yc _ (ELEM #bug#[ycoord] .A
      777777B)*4;
   % Return %
      RETURN(crdin);
   END.

(gcopy) PROCEDURE (delflg, grpflg, fig, atpt1, atpt2, from,
to);                                                          6C1F5
   %The figure of type fig displayed at screen location
   atpt1 is copied to "to" from coordinates "from" and the
   file is updated.  If delflg is true, the "from" figure
   is deleted.%
   LOCAL atadr, i, sgda, dgda, sstid, sdsp, dstid, ddsp,
   gi, op, atpt;
   REF atadr, sgda, dgda, gi;
   %---------------%
   gckemp(gdacds(to));
   IF grpflg THEN
      BEGIN
      gckgrp(atpt1, atpt2);
      giptrat();  %initialize attach table next entry
```

```
            pointer%
            &sgda _ gdacds(atpt1);
            gpsopn();
            WHILE (&atadr _ gnxtat()) DO
               IF (atadr.gatop=fig OR fig=figure) AND NOT
               (atadr.gatnew) THEN
                  IF ginside(atadr.gatcd, atpt1, atpt2) THEN
                     BEGIN
                     sstid _ atadr.gatstid;
                     sdsp _ atadr.gatdsp;
                     op _ atadr.gatop;
                     atpt _ atadr.gatcd;
                     &dgda _ gdacds(to);
                     gckopn(&dgda);
                     &gi _ gcopgi(sstid, sdsp, op, gridnc(&sgda,
                     &dgda, &atadr, from, to), dgda.glstid :
                     dstid, ddsp);
                     IF delflg THEN
                        BEGIN
                        gmvdel(&atadr, TRUE);
                        gdelgi(sstid, sdsp, op);
                        END;
                     IF op=annotation THEN
                        gmvtn(&dgda, dstid, dgda.gorg, TRUE)
                     ELSE
                        IF &gi THEN
                           gmvgi(&dgda, &gi, dgda.gorg, dstid,
                           ddsp, TRUE);
                     END;
            gpscls();
            %now clear the modify bits%
            giptrat();  %initialize attach table next entry
            pointer%
            WHILE (&atadr _ gnxtat()) DO
               BEGIN
               IF atadr.gatnew THEN atadr.gatnew _ FALSE;
               &atadr _ &atadr + gattel;
               END;
            RETURN;
            END
         ELSE
            BEGIN
            IF NOT &atadr _ gfind(fig, atpt1 : &sgda, sstid,
            sdsp) THEN
               err($"figure not found - gcopy");
            op _ atadr.gatop;
            &dgda _ gdacds(to);
            gckopn(&dgda);
            &gi _ gcopgi(sstid, sdsp, op, gridnc(&sgda, &dgda,
            &atadr,  from, to), dgda.glstid : dstid, ddsp);
            gpsopn();
            IF delflg THEN
               BEGIN
               gmvdel(&atadr, TRUE);
               gdelgi(sstid, sdsp, op);
               END;
```

```
            IF op=annotation THEN
               gmvtn(&dgda, dstid, dgda.gorg, FALSE)
            ELSE
               IF &gi THEN
                  gmvgi(&dgda, &gi, dgda.gorg, dstid, ddsp,
                  FALSE);
               gpscls();
               RETURN;
               END;
         END.
      (gcout) PROCEDURE (                                          6C1F6
         %Traces the diagram indicated onto an output device%
         node,             %Stid or Sid of the diagram%
         xl, xr, yt, yb,   %boundary in thousands - y positive
         down%
         rulflg,           %true if a boarder is to ruled around
         the figure%
         proc,             %address of the output routine to be
         used%
         whoflg,           %0=graphics subsystem,  1=output
         processor%
         rotflg);          %number of counterclockwise quadrant
         rotations%
         LOCAL savdev, savat, blkadr, stdb, stid;
         LOCAL locgda[38];  %must follow gdalgt%
         REF node, blkadr, prseqwk;
         %---------------------%
            CASE whoflg OF
               =0:  %graphics subsystem%
                  BEGIN
                  stid _ node;
                  IF NOT lodprop(stid, dhtyp : &blkadr, stdb)
                  THEN
                     BEGIN
                     %cleanup and exit%
                     IF NOT SKIP !closf(comjfn) THEN
                        err($"Bad trouble - take the day off");
                     reljfn(comjfn);
                     err($"diagram not found");
                     END;
                  savdev _ gdevtyp; gdevtyp _ vcom;
                  END;
               =1:  %output processor%
                  BEGIN
                  stid _ origin;
                  stid.stfile _ prseqwk.swcstid.stfile;
                  IF (stid _ lkupfast(stid, 0, &node, rsid)) =
                  endfil THEN
                     RETURN(FALSE);  %not found%
                  IF NOT lodprop(stid, dhtyp : &blkadr, stdb)
                  THEN
                     RETURN(FALSE);  %not found%
                  savdev _ gdevtyp; gdevtyp _ singer;
                  adoutl _ proc;
                  END;
               ENDCASE err(notyet);
```

```
      % setup a gda for the diagram trace out %
         locgda.gdstid _ stid;
         locgda.gtstid _ getitree(stdb);
         locgda.gcstid _ 0;
         locgda.glstid _ 0;
         locgda.gorg _ 0;
         locgda.gmleft   _ blkadr[sdbhdl].gmargleft;
         locgda.gmright  _ blkadr[sdbhdl].gmargright;
         locgda.gmtop    _ blkadr[sdbhdl].gmargtop;
         locgda.gmbottom _ blkadr[sdbhdl].gmargbottom;
         locgda.gwleft   _ float(gsigne(locgda.gmleft));
         locgda.gwright  _ float(gsigne(locgda.gmright));
         locgda.gwtop    _ float(gsigne(locgda.gmtop));
         locgda.gwbottom _ float(gsigne(locgda.gmbottom));
         locgda.gleft    _ xl;
         locgda.gright   _ xr;
         locgda.gtop     _ yb - yt;
         locgda.gbottom  _ 0;  %paper size%
         comxlfb _ xl;
         comxrfb _ xr;
         comytfb _ yt;
         comybfb _ yb;
         gsetpv($locgda);
         locgda.gwleft   _ float(gsigne(locgda.gmleft));
         locgda.gwright  _ float(gsigne(locgda.gmright));
         locgda.gwtop    _ float(gsigne(locgda.gmtop));
         locgda.gwbottom _ float(gsigne(locgda.gmbottom));
      % output the diagram %
         savat _ attach := 0;
         gdspdg($locgda);
         IF whoflg=0 THEN
            BEGIN
            gpsend(comnul); gpsend(comnul);
            gpsend(comeof);
            END;
         gdevtyp _ savdev;
         adout1 _ 0;
         attach _ savat;
      RETURN;
      END.

   (gdlfmt) PROCEDURE (ltvalu, intnst, brdnss);          6C1F7
      %Display line format%
      LOCAL lsvalu;
      LOCAL STRING str[50];
      CASE ltvalu OF
         = vcsolid:  *str* _ "line type solid";
         = vcdot:  *str* _ "line type dotted";
         = vcddash:  *str* _ "line type dot-dashed";
         = vcsdash:  *str* _ "line type short-dashed";
         = vcldash:  *str* _ "line type long-dashed";
         ENDCASE  *str* _ "invalid line type";
      gedsp($str);
      %find index in line style table for matching  intensity
      and broadness%
         lsvalu _ 0;
```

```
      UNTIL lsvalu = lstsiz OR (lstabl[lsvalu*2] = intnst
      AND lstabl[lsvalu*2+1] = brdnss) DO
         BUMP lsvalu;
   %index+1 is line style code%
      CASE lsvalu+1 OF
         = lsextra:  * str* _ "line style extra fine";
         = lsfine:  *str* _ "line style fine";
         = lsregular:  *str* _ "line style regular";
         = lsbold:  *str* _ "line style bold";
         = lsvbold:  *str* _ "line style very bold";
         ENDCASE  *str* _ "invalid line style";
   gedsp($str);
   RETURN;
   END.
(gdtfmt) PROCEDURE (gda, height, width, jvalue, fvalue,
svalue);                                                    6C1F8
   %Display text format%
   LOCAL j;
   LOCAL STRING str[50];
   REF gda;
   %character size%
      *str* _ "character height ";
      gfltxt($str, fdiv(float(height), gda.gdares));
      *str* _ *str*, " ", *[gbuabb]*, " width ";
      gfltxt($str, fdiv(float(width), gda.gdares));
      IF *[gbuabb]*="in" THEN
         BEGIN  %Give character size in points too%
         j _ grfix(fdiv(fdiv(float(height), gda.gdares),
         fpoint));
         *str* _ *str*, " in (", STRING(j), " pt)";
         gedsp($str);
         END
      ELSE guedsp($str);
   %justification%
      CASE jvalue OF
         =1:  %left%
            *str* _ "left justification";
         =2:  %right%
            *str* _ "right justification";
         =3:  %center%
            *str* _ "center justification";
         ENDCASE;
      gedsp($str);
   %font%
      CASE fvalue OF
         =fntc:  *str* _ "Courier font";
         =fntd:  *str* _ "Directory font";
         =fntf:  *str* _ "Film font";
         =fnto:  *str* _ "ORCB font";
         =fntnm:  *str* _ "NMAMIcrofont font";
         =fntn:  *str* _ "NewsGothic font";
         =fntt:  *str* _ "TimesRoman font";
         ENDCASE *str* _ "Invalid font";
      gedsp($str);
   %emphasis%
      CASE svalue.gempb OF
```

```
            =0:  *str* _ "Medium boldness";
            =1:  *str* _ "Light typeface";
            =2:  *str* _ "Bold typeface";
            ENDCASE *str* _ "Invalid typeface";
         gedsp($str);
         IF svalue.gemps
            THEN *str* _ " - Slanted"
            ELSE *str* _ " - Straight";
         gedsp($str);
         IF svalue.gempu
            THEN *str* _ " - Underlined"
            ELSE *str* _ " - Not underlined";
         gedsp($str);
         IF svalue.gempm
            THEN *str* _ " - Monospaced"
            ELSE *str* _ " - Proportional spacing";
         gedsp($str);
      RETURN;
      END.
   (gerase) PROCEDURE (grpflg, fig, atpt1, atpt2);          6C1F9
      %The figure of type fig displayed at screen location
      atpt1 is deleted and marked.%
      LOCAL atadr, gda, stid, dsp, op, i;
      REF atadr, gda;
      %---------------%
      gckemp(gdacds(atpt1));
      IF grpflg THEN
         BEGIN
         gckgrp(atpt1, atpt2);
         gpsopn();
         giptrat();  %initialize attach table next entry
         pointer%
         WHILE (&atadr _ gnxtat()) DO
            IF atadr.gatop=fig OR fig=figure THEN
               IF ginside(atadr.gatcd, atpt1, atpt2) THEN
                  BEGIN
                  &gda _ gdacds(atadr.gatcd);
                  stid _ atadr.gatstid;
                  dsp _ atadr.gatdsp;
                  op _ atadr.gatop;
                  gmvdel(&atadr, TRUE);
                  gdelgi(stid, dsp, op);
                  END;
         gpscls();
         RETURN;
         END
      ELSE
         BEGIN
         IF NOT &atadr _ gfind(fig, atpt1 : &gda, stid, dsp)
         THEN  err($"figure not found - gerase");
         gpsopn();
         op _ atadr.gatop;
         gmvdel(&atadr, TRUE);
         gdelgi(stid, dsp, op);
         gpscls();
         RETURN;
```

```
      END;
   END.
(gmakewindow)   % LB ;  make initial graphics window %
PROCEDURE;                                                    6C1F10
   % Procedure description
      FUNCTION
         Allocate and initialize a window area (BE display
         structure) for the initial graphics window.  Do a
         FE create window command.  Store the returned
         window id in the allocated window area.
      ARGUMENTS
         none
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         Sets gprwndw
      %
   % Declarations %
      REF gprwndw, prwndw;
      LOCAL LIST retlist[1];
      REF gcommands;
   % Set up window area %
      &gprwndw _ newwa();  %allocate window area%
      %defaults from primary window%
         gprwndw.wiatt _ prwndw.wiatt;
         gprwndw.wiselector _ prwndw.wiselector;
      gprwndw.widowner _ prwndw.widindex;  %user primary
      window as owning%
      gprwndw.wtype _ gwtype;  %graphics window type%
      gprwndw.wipriority _ 2;  %allows other windows to
      overlap%
      gprwndw.wuplx _ scnleft;  %window left boundary%
      gprwndw.wuply _ scntop;  %window top boundary%
      gprwndw.wlrx _ scnright;  %window right boundary%
      gprwndw.wlry _ scnbot;  %window bottom boundary%
   % Create graphics window in  FE %
      #gcommands# _ ;
      bldcw(&gprwndw, &gcommands);  % build FE command %
      prcmds(FALSE, &gcommands, FALSE, $retlist);  % send
      to FE %
      gprwndw.widindex _ ELEM #retlist#[1];  % store FE wid
      %
      #gcommands# _ ;
   % Return %
      RETURN;
   END.

(gmove) PROCEDURE (grpflg, fig, atpt1, atpt2, from, to);
                                                              6C1F11
   %The figure of type fig displayed at screen location
   atpt1 is copied to "to" from coordinates "from" and the
   figure at the "from" coordinates is then deleted.  Both
   files are updated.%
   LOCAL atadr, i, gda, stid, dsp, gi, op, atpt;
```

```
      REF atadr, gda, gi;
      %---------------%
      IF gdacds(from) = gdacds(to) THEN
         gtrnsl(grpflg, fig, atpt1, atpt2, from, to)
      ELSE
         gcopy(TRUE, grpflg, fig, atpt1, atpt2, from, to);
      RETURN;
      END.
   (gntwnd) PROCEDURE (tfold, tfnew, stid);                6C1F12
      %Calculate new text window if text will not fit in old
      window%
      LOCAL TEXT POINTER tpf, tpe;
      LOCAL nlines, cpline, dffrn, cds;
      REF tfold, tfnew;
      IF tfnew.gtfch > tfold.gtfch THEN
         BEGIN
         %Set new window keeping same number of lines%
            cpline _ (tfold.gtfwdt - 2*tfold.gtfbdr) /
            tfold.gtfcw;
            nlines _ (tfold.gtfhgt - 2*tfold.gtfbdr) /
            tfold.gtfch;
            tfnew.gtfwdt _ (2*tfnew.gtfbdr) +
            (cpline*tfnew.gtfcw);
            tfnew.gtfhgt _ (2*tfnew.gtfbdr) +
            (nlines*tfnew.gtfch);
         %Compare old and new widths and extend rectangle
         according to justification, i.e. change upper left
         coordinate%
            dffrn _ tfnew.gtfwdt - tfold.gtfwdt;
            cds _ tfold.gtfcds;
            CASE tfnew.gtfjus OF
               =1:  %left%
                  NULL;  %extend right%
               =2:  %right%
                  cds.xc _ cds.xc - dffrn;  %extend left%
               =3:  %center%
                  cds.xc _ cds.xc - dffrn/2;  %extend both%
               ENDCASE;
            tfnew.gtfcds _ cds;
         END;
      RETURN;
      END.
   (gsdrws) PROCEDURE (gda);                               6C1F13
      %Show drawspecs%
      LOCAL STRING str[50];
      REF gda;
      gfbctl(typenullit);  %clear text display screen%
      %grid%
         *str* _ "grid ";
         gfltxt($str, fdiv(gda.gdagrd, gda.gdares));
         guedsp($str);
      %resolution%
         *str* _ "resolution ";
         gfltxt($str, gda.gdares);
         *str* _ *str*, " increments per";
         guedsp($str);
```

```
        %page size%
           *str* _ "page size ";
           gfltxt($str, fdiv(float(absdif(gda.gmright,
           gda.gmleft)), gda.gdares));
           *str* _ *str*, " ", *[gbuabb]*, " by ";
           gfltxt($str, fdiv(float(absdif(gda.gmtop,
           gda.gmbottom)), gda.gdares));
           guedsp($str);
        %text format%
           gdtfmt($gda, gda.gdach, gda.gdacw, gda.gdajus,
           gda.gdafnt, gda.gdaemp);
        %line type and style%
           gdlfmt(gda.gdalvec, gda.gdalint, gda.gdalbrd);
        %arrowhead size%
           *str* _ "arrowhead height ";
           gfltxt($str, fdiv(float(gda.gdaahgt), gda.gdares));
           *str* _ *str*, " ", *[gbuabb]*, " width ";
           gfltxt($str, fdiv(float(gda.gdaawdt*2), gda.gdares));
           guedsp($str);
        gfbctl(addcalit, $str);  %wait for CA and then refresh
        screen%
        RETURN;
        END.
     (gsetas) PROCEDURE (gda, param, param2);                6C1F14
        %Return arrowhead height and width%
        REF param, param2, gda;
        RETURN(grfix(gtxtpg(&param, &gda)),
        grfix(gtxtpg(&param2, &gda)) / 2);
        END.
     (gsetcs) PROCEDURE (gda, param);                        6C1F15
        %Return character height and width%
        LOCAL j;
        REF param, gda;
        j _ grfix(gtxtpg(&param, &gda));
        RETURN(j, fix(fdiv(float(j), aspctr)));
        END.
     (gsetf) PROCEDURE (param);                              6C1F16
        %Return font code%
        LOCAL j;
        LOCAL STRING str[1];
        CASE param OF
           =1:  %Courier font%
              j _ fntc;
           =2:  %Directory font%
              j _ fntd;
           =3:  %Film%
              j _ fntf;
           =4:  %ORCR%
              j _ fnto;
           =5:  %NMAMIcrofont%
              j _ fntnm;
           =6:  %NewsGothic%
              j _ fntn;
           =7:  %TimesRoman%
              j _ fntt;
           ENDCASE err(notyet);
```

```
    RETURN(j);
    END.
(gsgrid) PROCEDURE (gda, scd1, scd2);                          6C1F17
    %Display the current grid points within the given
    window.%
    LOCAL x, y, x1, y1, x2, y2, gi[2];
    REF gda;
    x _ gridxy(&gda, scd1.xc, scd1.yc);
    y _ gridxy(&gda, scd2.xc, scd2.yc);
    x1 _ MIN(gngrid(x.xc, &gda), gngrid(y.xc, &gda));
    x2 _ MAX(gngrid(x.xc, &gda), gngrid(y.xc, &gda));
    y1 _ MAX(gngrid(x.yc, &gda), gngrid(y.yc, &gda));
    y2 _ MIN(gngrid(x.yc, &gda), gngrid(y.yc, &gda));
    gi _ 0;
    gi.gphops _ point;
    gi.gphlgt _ 2;
    y _ y1;
    gpsopn();
    DO
       BEGIN
       gi[1].yc _ gcdgrd(y, &gda);
       x _ x1;
       DO
          BEGIN
          gi[1].xc _ gcdgrd(x, &gda);
          gdspgi(&gda, $gi, 0, FALSE, 0, 0); %doesn't use
          attaching table%
          BUMP x;
          END
       UNTIL x > x2 OR inptrf;
       BUMP DOWN y;
       END
    UNTIL y < y2 OR inptrf;
    gpscls();
    RETURN;
    END.
(gsunit) PROCEDURE (gda);                                      6C1F18
    %Show unit%
    LOCAL addr, i;
    LOCAL STRING str[50];
    REF gda, addr, gunitb;
    gfbctl(typenullit);  %clear text display screen%
    %Basic unit%
       *str* _ "Basic unit ", *[gbuabb]*, " with ";
       gfltxt($str, gda.gdares);
       *str* _ *str*, " increments per";
       guedsp($str);
    %Units%
       &addr _ &gunitb;
       FOR i _ 0 UP UNTIL =guntcl DO
          BEGIN
          IF *[&addr]* # *[gbuabb]* THEN
             BEGIN
             *str* _ *[&addr]*, " ";
             gfltxt($str, addr.gutpsi);
             guedsp($str);
```

```
            END;
         &addr _ &addr + guntel;  %go to next entry%
         END;
      gfbctl(addcalit, $str);  %wait for CA and refresh
      screen%
      RETURN;
      END.
   (gtfit) PROCEDURE (tp1, tp2);                          6C1F19
      %Given tp1 and tp2 find the width and depth of a text
      string.%
      LOCAL nlines, nchars, mchars, i;
      LOCAL STRING str[2000];
      REF tp1, tp2;
      *str* _ tp1 tp2;
      nlines _ 0;  nchars _ 0;  mchars _ 0;
      IF str.L > 0 THEN nlines _ 1;
      FOR i_1 UP UNTIL >str.L DO
         CASE *str*[i] OF
            =SP, =PT:
               BEGIN
               BUMP nchars;
               mchars _ MAX(mchars, nchars);
               END;
            =NP:
               BEGIN
               BUMP nlines;
               nchars _ 0;
               END;
            ENDCASE;
      RETURN(mchars, nlines);
      END.
   (gtrnsl) PROCEDURE (grpflg, fig, atpt1, atpt2, from, to);
                                                          6C1F20
      %The figure of type fig displayed at screen location
      atpt1 is translated to "to" from location "from" and the
      file is updated.%
      LOCAL atadr, i, gda, stid, dsp, gi, op, atpt;
      REF atadr, gda, gi;
      %---------------%
      gckemp(gdacds(to));
      IF grpflg THEN
         BEGIN
         gckgrp(atpt1, atpt2);
         giptrat();  %initialize attach table next entry
         pointer%
         gpsopn();
         WHILE (&atadr _ gnxtat()) DO
            IF (atadr.gatop=fig OR fig=figure) AND NOT
            (atadr.gatnew) THEN
               IF ginside(atadr.gatcd, atpt1, atpt2) THEN
                  BEGIN
                  &gda _ gdacds(atadr.gatcd);
                  stid _ atadr.gatstid;
                  dsp _ atadr.gatdsp;
                  op _ atadr.gatop;
                  atpt _ atadr.gatcd;
```

```
                        &gi _ gmovgi(stid, dsp, op, gridnc(&gda,
                        &gda, &atadr, from, to));
                        gmvdel(&atadr, TRUE);
                        IF op=annotation THEN
                           gmvtn(&gda, stid, gclorg(stid), TRUE)
                        ELSE
                           IF &gi THEN
                              gmvgi(&gda, &gi, gclorg(stid), stid,
                              dsp, TRUE);
                        END;
                  gpscls();
                  %now clear the modify bits%
                  giptrat();
                  WHILE (&atadr _ gnxtat()) DO
                     IF atadr.gatnew THEN atadr.gatnew _ FALSE;
                  RETURN;
                  END
               ELSE
                  BEGIN
                  IF NOT &atadr _ gfind(fig, atpt1 : &gda, stid, dsp)
                  THEN
                     err($"figure not found - gmove");
                  gpsopn();
                  op _ atadr.gatop;
                  &gi _ gmovgi(stid, dsp, op, gridnc(&gda, &gda,
                  &atadr, from, to));
                  gmvdel(&atadr, TRUE);
                  IF op=annotation THEN
                     gmvtn(&gda, stid, gclorg(stid), FALSE)
                  ELSE
                     IF &gi THEN
                        gmvgi(&gda, &gi, gclorg(stid), stid, dsp,
                        FALSE);
                  gpscls();
                  RETURN;
                  END;
               END.
            %SPL routines%
               (splrd) PROCEDURE (gda);                                  6C1G1
                  %In put a single block form the SPL file to the diagram
                  in the gda.%
                  REF gda;
                  LOCAL char, value, se, msg, x1, y1, x2, y2, cd1, cd2,
                  stid, dsp, gi;
                  LOCAL TEXT POINTER sftp, setp;
                  LOCAL STRING locstr[100];
                  gckopn(&gda);
                  msg _ 0;
                  LOOP
                     BEGIN
                     char _ splnc();
                     CASE char OF
                        ="M, ="m :  %Margin command%
                           BEGIN
                           se _ FALSE;
                           IF splrn( : value) THEN gda.gmleft _ value
```

```
            ELSE se _ TRUE;
         IF splrn( : value) THEN gda.gmright _ value
            ELSE se _ TRUE;
         IF splrn( : value) THEN gda.gmtop _ value
            ELSE se _ TRUE;
         IF splrn( : value) THEN gda.gmbottom _ value
            ELSE se _ TRUE;
         IF gda.gmright-gda.gmleft<=0 THEN
            BEGIN
            se _ TRUE;
            gda.gmleft _ (gda.gmright := gda.gmleft);
            END;
         IF gda.gmtop-gda.gmbottom<=0 THEN
            BEGIN
            se _ TRUE;
            gda.gmtop _ (gda.gmbottom := gda.gmtop);
            END;
         gda.gwleft   _ float(gsigne(gda.gmleft));
         gda.gwright  _ float(gsigne(gda.gmright));
         gda.gwtop    _ float(gsigne(gda.gmtop));
         gda.gwbottom _ float(gsigne(gda.gmbottom));
         grpdhd(&gda);  %Reprop diagram header block%
         splfel();
         IF se THEN msg _ msg .V sple01;
         END;
      ='L, ='l :  %Line commmand%
         BEGIN
         se _ FALSE;
         IF splrn( : value) THEN x1 _ value
            ELSE se _ TRUE;
         IF splrn( : value) THEN y1 _ value
            ELSE se _ TRUE;
         IF splrn( : value) THEN x2 _ value
            ELSE se _ TRUE;
         IF splrn( : value) THEN y2 _ value
            ELSE se _ TRUE;
         IF NOT se THEN
            BEGIN
            cd1.xc _ x1;  cd1.yc _ y1;
            cd2.xc _ x2;  cd2.yc _ y2;
            gi _ gbldgi(&gda, line, 0, cd1, cd2, 0 :
            stid, dsp);
            END;
         LOOP
            BEGIN
            IF splrn( : value) THEN cd1.xc _ value
               ELSE EXIT LOOP;
            IF splrn( : value) THEN cd1.yc _ value
               ELSE BEGIN se _ TRUE; EXIT LOOP;  END;
            gi _ gexlgi(stid, dsp, 20 %extend%, cd1,
            gda.glstid : stid, dsp);
            END;
         splfel();
         END;
      ='T, ='t :  %text commmand%
         BEGIN
```

```
                se _ FALSE;
                IF splrn( : value) THEN x1 _ value
                   ELSE se _ TRUE;
                IF splrn( : value) THEN y1 _ value
                   ELSE se _ TRUE;
                IF splrn( : value) THEN y2 _ value
                   ELSE se _ TRUE;
                IF splrs($locstr) THEN
                   BEGIN
                   setp _ sftp _ $locstr;
                   setp.stastr _ sftp.stastr _ TRUE;
                   sftp[1] _ 1;
                   setp[1] _ locstr.L + 1;
                   END
                   ELSE se _ TRUE;
                IF NOT se THEN
                   BEGIN
                   cd1.xc _ x1;  cd1.yc _ y1;
                   gda.gdacw _ (12*y2)/17;
                   gda.gdach _ y2;
                   gda.gdabdr _ 0;
                   gda.gdaemp _ 0;
                   gda.gdafnt _ 0;
                   gda.gdajus _ 1;
                   gda.gdadm3 _ 0;
                   x2 _ (gda.gdacw)*(locstr.L + 1);
                   gbldtn(&gda, cd1, x2, y2, $sftp, $setp);
                   END;
                splfel();
                END;
             =SP :  %Ignore%  NULL;
             =0  :  %End of file%  RETURN(msg);
             ="# :  %End of block%  RETURN(msg);
             ENDCASE
                BEGIN
                msg _ msg .V sple02;
                splfel();
                END;
          END;
       END.
(splfb) PROCEDURE (n);                                          6C1G2
       %Position the SPL file byte pointer the the text of the
       data block.  If the block is not found return false.%
       LOCAL char, value;
       IF NOT SKIP !sfptr(spljfn, 0) THEN
          err($"Cannot set file pointer");
       LOOP
          BEGIN
          IF NOT char_splnc() THEN RETURN (FALSE);
          IF char="# THEN
             BEGIN
             spleol _ FALSE;
             IF splrn( : value) THEN
                IF n=value THEN
                   BEGIN
                   splfel();
```

```
                    RETURN (TRUE);
                    END;
                END;
            END;
        END.
    (splrn) PROCEDURE;                                          6C1G3
        %Return the value of the next number in the SPL file.
        Return true if ok else false.%
        LOCAL cp, char, value, sgn;
        LOCAL STRING valstr[200];
        IF spleol THEN RETURN (FALSE);
        cp _ 0;
        sgn _ 1;
        LOOP
            BEGIN
            char _ splnc();
            IF NOT char THEN RETURN (FALSE);
            IF char IN['0, '9] THEN EXIT LOOP;
            IF char='- THEN sgn _ -1;
            IF char='; THEN RETURN (FALSE);
            END;
        cp _ 1;
        *valstr*[cp] _ char;
        LOOP
            BEGIN
            char _ splnc();
            IF char IN['0, '9] THEN
                BEGIN
                BUMP cp;
                IF cp> 200 THEN RETURN (FALSE);
                *valstr*[cp] _ char;
                END
            ELSE EXIT LOOP;
            END;
        value _ VALUE($valstr) * sgn;
        IF char='. THEN
            %skip over the fraction%
            UNTIL NOT splnc() IN['0, '9] DO NULL;
        RETURN (TRUE, value);
        END.
    (splrs) PROCEDURE(str);                                     6C1G4
        %Return the value of the next string in the SPL file.
        Return true if ok else false.%
        LOCAL cp, char, value, sgn;
        REF str;
        IF spleol THEN RETURN (FALSE);
        str.L _ 0;
        cp _ 0;
        sgn _ 1;
        LOOP
            BEGIN
            char _ splnc();
            IF NOT char THEN RETURN (FALSE);
            IF char='" THEN EXIT LOOP;
            IF char='; THEN RETURN (FALSE);
            END;
```

```
        LOOP
            BEGIN
            char _ splnc();
            IF NOT char="" THEN
                BEGIN
                BUMP cp;
                IF cp>str.M THEN RETURN (TRUE);
                *str*[cp] _ char;
                END
            ELSE
                BEGIN
                char _ splnc();
                IF char="" THEN
                    BEGIN
                    BUMP cp;
                    IF cp>str.M THEN RETURN (TRUE);
                    *str*[cp] _ "";
                    END
                ELSE EXIT LOOP;
                END;
            END;
        RETURN (TRUE, value);
        END.
    (splnc) PROCEDURE;                                          6C1G5
        %Get the next character from the SPL file.  EOF returns
        false.%
        LOCAL char;
        !bin(spljfn);  char _ R2;
        !gtsts(spljfn);
        IF R2 .A 001000000000B THEN RETURN (FALSE);
        CASE char OF
            =EOL, =CR, =LF : RETURN(" ");
            =";" :
                BEGIN
                spleol _ TRUE;
                RETURN(char);
                END;
            =0 : RETURN(" ");
            ="% :
                BEGIN
                % skip comments %
                DO
                    BEGIN
                    !bin(spljfn);  char _ R2;
                    !gtsts(spljfn);
                    IF R2 .A 001000000000B THEN RETURN (FALSE);
                    END
                UNTIL char="%;
                !bin(spljfn);  char _ R2;
                !gtsts(spljfn);
                IF R2 .A 001000000000B THEN RETURN (FALSE);
                REPEAT CASE;
                END;
            ENDCASE RETURN(char);
        END.
    (splfel) PROCEDURE;                                         6C1G6
```

```
        %Find the next ";"%
        LOCAL char;
        IF spleol THEN
           BEGIN
           spleol _ FALSE;
           RETURN;
           END;
        LOOP
           BEGIN
           char _ splnc();
           IF char=';' OR char=0 THEN
              BEGIN
              spleol _ FALSE;
              RETURN;
              END;
           END;
        END.
%FINISH.%%% % end of file 1 of support routines for split
```

```
version of subsystem%
%Graphics multi-viewport manipulation%
   (gmvdel) PROCEDURE (delfig , mflag);                          6C1I1
      %delete all instances of figure except new one -- delete
      attach table entry and mark figure with an X%
         LOCAL atadr, i, dsp, stid;
         REF atadr, delfig, attach;
         stid _ delfig.gatstid;
         dsp _ delfig.gatdsp;
         &atadr _ &attach + gatfst;
         %search attach table for figure%
         FOR i_1 UP UNTIL >attach[gatcl] DO
            BEGIN
            IF (atadr.gatstid = stid) AND (atadr.gatdsp = dsp)
            AND NOT atadr.gatnew THEN
               gdelat(&atadr, mflag);
            &atadr _ &atadr + gattel;
            END;
      RETURN;
      END.
   (gmvgi) PROCEDURE (gda, gi, org, lstid, dsp, nflag);    6C1I2
      %Portray all instances of graphics instruction.  check
      instnc viewports for diagram portrayed in gda%
      LOCAL instnc;
      REF gda, gi, instnc, gdabas;
      &instnc _ &gdabas;
      DO
      IF instnc.gdstid = gda.gdstid THEN
         gdspgi(&instnc, &gi, org, lstid, dsp, nflag)
      UNTIL NOT (&instnc _ instnc.gdanxt);
      RETURN;
      END.
   (gmvtn) PROCEDURE (gda, lstid, org, nflag);                   6C1I3
      %Portray all instances of text.  check instnc viewports
      for diagram portrayed in gda%
      LOCAL instnc;
      REF gda, instnc, gdabas;
      &instnc _ &gdabas;
      DO
      IF instnc.gdstid = gda.gdstid THEN
         gdsptn(&instnc, lstid, org, nflag)
      UNTIL NOT (&instnc _ instnc.gdanxt);
      RETURN;
      END.
%Graphics portrayal system%
   (grecred) PROCEDURE;                                          6C1J1
      %Recreate all graphics da's%
      LOCAL gda;
      LOCAL ndrw;
      REF gda, gdabas;
      %---------------%
      IF NOT &gdabas THEN err($"graphics system off");
      &gda _ &gdabas;
      gclrat();
      gclr();
      IF logof THEN goutsn();
```

```
        DO
           BEGIN
           goutln(&gda);
           IF gda.gdstid THEN  %zero if new viewport%
           BEGIN
              gsetpv(&gda);
              goutmgn(&gda);
              FOR ndrw_0 UP 2 UNTIL >= gblrrc DO
                 BEGIN
                 gxblrr _ gtblrr[ndrw];
                 gyblrr _ gtblrr[ndrw+1];
                 gdspdg(&gda);
                 END;
           END;
           &gda _ gda.gdanxt;
           END
        UNTIL NOT &gda;
        RETURN;
        END.
     (gdspdg) PROCEDURE (gda);                                    6C1J2
        %Display a diagram inferior gree.%
        LOCAL stid, stdb, blkadr, lstid, org;
        REF gda, blkadr;
        IF NOT gda.gtstid THEN err($"tree not found - gdspdg");
        stid _ getsub(gda.gtstid);
        gpsopn();
        UNTIL stid=gda.gtstid OR stid=endfil OR inptrf DO
           BEGIN
           IF lodprop(stid, chtyp : &blkadr, stdb) THEN
              IF lstid _ getitree(stdb) THEN
                 gdspcl(&gda, lstid, blkadr[sdbhdl].gcorg);
           stid _ getnxt(stid);
           END;
        gpscls();
        RETURN;
        END.
     (gdspcl) PROCEDURE (gda, lstid, org);                        6C1J3
        %Display the cell inferior tree.%
        LOCAL stid, stdb, blkadr, blktop, endadr;
        REF gda, blkadr, blktop;
        stid _ getsub(lstid);
        UNTIL stid=lstid OR stid=endfil OR inptrf DO
           BEGIN
           IF lodprop(stid, lwtyp : &blkadr, stdb) THEN
              BEGIN
              &blkadr _ &blktop _ &blkadr + sdbhdl;
              endadr _ &blkadr + getdblen(stdb) - sdbhdl;
              WHILE &blkadr<endadr AND NOT inptrf DO
                 BEGIN
                 gdspgi(&gda, &blkadr, org, stid,
                 &blkadr-&blktop, FALSE);
                 &blkadr _ &blkadr + blkadr.gphlgt;
                 END;
              END
           ELSE
              BEGIN
```

```
            gdsptn(&gda, stid, org, FALSE);
            END;
         stid _ getsuc(stid);
         END;
      RETURN;
      END.
   (gdspgi) PROCEDURE (gda,gins,org,stid,dsp,newflg);     6C1J4
      %Draw the line work indicated by the graphics
      instruction addressed by gins on the display using the
      transformation within the gda%
      LOCAL i,j,x,y,fst,pair[6],spair[2];
      LOCAL xcnt, ycnt, angle, radius, inca, scw;
      REF gins,gda;
      %---------------%
      atvisible _ FALSE;  %set up to compute attaching point%
      atch.xc _ 77777B;
      atch.yc _ 0;
      fst _ 1;  %index fo first data element of gi%
      clast _ -1;
      gcurvc_gins.gphvec;  %line style data%
      gcurin_gins.gphint;
      gcurbd_gins.gphbrd;
      devset(&gda);
      CASE gins.gphops OF
         =point:  %point%
            BEGIN
            grptofp(gins[fst], $pair, org);
            gline(&gda, pair   , pair[1], pair   , pair[1]);
            END;
         =line:  %line word graphics instruction%
            BEGIN
            grptofp(gins[fst],$pair,org);
            org _ ghadd(gcftoi($pair), org);
            j _ 1;
            FOR i_fst+1 UP UNTIL >=gins.gphlgt DO
               BEGIN
               IF gins[i]=gpenup THEN j _ 0
               ELSE
                  BEGIN
                  grptofp(gins[i],$pair[2],org);
                  BUMP j;
                  IF j=2 THEN
                     BEGIN
                     gline(&gda, pair, pair[1], pair[2],
                     pair[3]);
                     j _ 1;
                     END;
                  pair    _ pair[2];
                  pair[1] _ pair[3];
                  END;
               END;
            END;
         =rectangle:  %rectangle%
            BEGIN
            grptofp(gins[fst], $pair, org);
            spair    _ float(gins[fst+1].xc);
```

```
            spair[1] _ float(ginst[fst+1].yc);
            pair[2] _ fadd(pair, spair);
            pair[3] _ fsub(pair[1], spair[1]);
            gline(&gda, pair   , pair[1], pair[2], pair[1]);
            gline(&gda, pair[2], pair[1], pair[2], pair[3]);
            gline(&gda, pair[2], pair[3], pair   , pair[3]);
            gline(&gda, pair, pair[3], pair, pair[1]);
            END;
         =diamond:  %diamond%
            BEGIN
            grptofp(ginst[fst], $pair, org);
            spair    _ float(ginst[fst+1].xc);
            spair[1] _ float(ginst[fst+1].yc);
            pair[2] _ fadd(pair, spair);
            pair[3] _ fsub(pair[1], spair[1]);
            gline(&gda, pair   , pair[1], pair[2], pair[3]);
            pair[3] _ fadd(pair[1], spair[1]);
            gline(&gda, pair   , pair[1], pair[2], pair[3]);
            pair _ fadd(pair[2], spair);
            gline(&gda, pair   , pair[1], pair[2], pair[3]);
            pair[3] _ fsub(pair[1], spair[1]);
            gline(&gda, pair   , pair[1], pair[2], pair[3]);
            END;
         =itriangle:  %itriangle%
            BEGIN
            grptofp(ginst[fst], $pair[2], org);
            spair    _ float(ginst[fst+1].wdt);
            spair[1] _ float(ginst[fst+1].hgt);
            CASE ginst[fst+1].dir OF
               =north:
                  BEGIN
                  pair    _ fsub(pair[2], spair   );
                  pair[1] _ pair[5] _ fsub(pair[3], spair[1]);
                  pair[4] _ fadd(pair[2], spair   );
                  END;
               =south:
                  BEGIN
                  pair    _ fsub(pair[2], spair   );
                  pair[1] _ pair[5] _ fadd(pair[3], spair[1]);
                  pair[4] _ fadd(pair[2], spair   );
                  END;
               =east:
                  BEGIN
                  pair[1] _ fsub(pair[3], spair   );
                  pair    _ pair[4] _ fsub(pair[2], spair[1]);
                  pair[5] _ fadd(pair[3], spair   );
                  END;
               =west:
                  BEGIN
                  pair[1] _ fsub(pair[3], spair   );
                  pair    _ pair[4] _ fadd(pair[2], spair[1]);
                  pair[5] _ fadd(pair[3], spair   );
                  END;
               ENDCASE err($"Invalid direction code -
               gdspgi");
            gline(&gda, pair   , pair[1], pair[2], pair[3]);
```

```
            gline(&gda, pair[2], pair[3], pair[4], pair[5]);
            gline(&gda, pair[4], pair[5], pair   , pair[1]);
            END;
         =arrowhead:  %arrowhead%
            BEGIN
            grptofp(ginst[fst], $pair[2], org);
            spair    _ float(ginst[fst+1].wdt);
            spair[1] _ float(ginst[fst+1].hgt);
            CASE ginst[fst+1].dir OF
               =north:
                  BEGIN
                  pair    _ fsub(pair[2], spair   );
                  pair[1] _ pair[5] _ fsub(pair[3], spair[1]);
                  pair[4] _ fadd(pair[2], spair   );
                  END;
               =south:
                  BEGIN
                  pair    _ fsub(pair[2], spair   );
                  pair[1] _ pair[5] _ fadd(pair[3], spair[1]);
                  pair[4] _ fadd(pair[2], spair   );
                  END;
               =east:
                  BEGIN
                  pair[1] _ fsub(pair[3], spair   );
                  pair    _ pair[4] _ fsub(pair[2], spair[1]);
                  pair[5] _ fadd(pair[3], spair   );
                  END;
               =west:
                  BEGIN
                  pair[1] _ fsub(pair[3], spair   );
                  pair    _ pair[4] _ fadd(pair[2], spair[1]);
                  pair[5] _ fadd(pair[3], spair   );
                  END;
               ENDCASE err($"Invalid direction code -
               gdspgi");
            gline(&gda, pair   , pair[1], pair[2], pair[3]);
            gline(&gda, pair[2], pair[3], pair[4], pair[5]);
            END;
         =circle:  %circle%
            BEGIN
            grptofp(ginst[fst], $pair, org);
            xcnt _ pair;
            ycnt _ pair[1];
            radius _ float(ginst[fst+1].yc); %radius%
            scw _ fix(fmul(gda.gvtrn, radius));
            CASE scw OF
               <10: inca _ 360;
               <20: inca _ 90;
               <30: inca _ 45;
               <40: inca _ 30;
               <50: inca _ 15;
               <60: inca _ 10;
               <100: inca _ 5;
               ENDCASE inca _ 3;
            angle _ 0;
            pair    _ fadd(xcnt,fmul(radius,fcos(angle)));
```

```
            pair[1] _ fadd(ycnt,fmul(radius,fsin(angle)));
            FOR i_inca UP inca UNTIL >360 DO
               BEGIN
               angle _ fdrad(float(i));
               pair[2] _ fadd(xcnt,fmul(radius,fcos(angle)));
               pair[3] _ fadd(ycnt,fmul(radius,fsin(angle)));
               gline(&gda, pair   , pair[1], pair[2],
               pair[3]);
               pair    _ pair[2];
               pair[1] _ pair[3];
               END;
            END;
         ENDCASE;
      IF atvisible AND stid THEN
         BEGIN
         gputat(atch, gins.gphops, stid, dsp, newflg);
         END;
      RETURN;
      END.
   (gdsptn) PROCEDURE (gda,stid,org,newflg);                    6C1J5
      %Display the text in the branch at stid in the the text
      window in the graphics text format property using the
      page origin given and the viewing transformation given
      in the gda.%
      LOCAL xl, xr, yt, yb, y, n, fst, lst, first, blkadr,
      stdb, pair[2], i;
      LOCAL STRING strng[2000];
      REF blkadr, gda;
      %---------------%
      atvisible _ FALSE;  %set up to compute attaching point%
      atch.xc _ 777777B;
      atch.yc _ 0;
      IF NOT lodprop(stid, gtftyp : &blkadr, stdb)
         THEN err($"gtftyp not found - gdsptn");
      FOR i_0 UP UNTIL =5 DO
         gcurtf[i] _ blkadr[sdbhdl+i];
      devset(&gda);
      xl _ gsigne(gcurtf.gtfcds.xc) + gcurtf.gtfbdr;
      xr _ xl + gcurtf.gtfwdt - (2 * gcurtf.gtfbdr);
      yt _ gsigne(gcurtf.gtfcds.yc);
      y _  yt - gcurtf.gtfbdr - gcurtf.gtfch;
      yb _ yt - gcurtf.gtfhgt + gcurtf.gtfbdr;
      n _ (xr-xl)/gcurtf.gtfcw;
      IF NOT n IN [1,132] THEN
         BEGIN
         %something is wrong%
         RETURN;
         END;
      *strng* _ SF(stid) SE(stid);
      i _ 1;
      LOOP
         BEGIN
         %find the first printing%
         LOOP
            BEGIN
            fst _ i;
```

```
            IF i>strng.L THEN EXIT LOOP 2;
            IF *strng*[i]=PT THEN EXIT LOOP;
            BUMP i;
            END;
         %find n characters%
         lst _ FALSE;
         LOOP
            BEGIN
            BUMP i;
            IF i>strng.L THEN
               BEGIN
               lst _ i-1;
               EXIT LOOP;
               END;
            CASE *strng*[i] OF
               =SP:  %word break%
                  IF i-fst<n THEN
                     lst _ i-1
                  ELSE
                     EXIT LOOP;
               =NP:  %line break%
                  BEGIN
                  lst _ i-1;
                  EXIT LOOP;
                  END;
               =PT:   %printing%
                  IF i-fst+1>=n THEN
                     BEGIN
                     IF lst THEN
                        BEGIN
                        IF i=strng.L THEN
                           BEGIN lst_i; EXIT LOOP; END;
                        IF *strng*[i+1]=NP THEN
                           BEGIN lst_i; BUMP i; EXIT LOOP;
                           END;
                        IF *strng*[i+1]=SP THEN
                           BEGIN lst_i; EXIT LOOP; END;
                        i _ lst;
                        EXIT LOOP;
                        END
                     ELSE
                        BEGIN
                        lst _ i;
                        EXIT LOOP;
                        END;
                     END;
               ENDCASE;
            END;
         UNTIL *strng*[lst]=PT DO BUMP DOWN lst;
         gtline(&gda, org, xl, xr, y, $strng, fst, lst);
         CASE gdevtyp OF
            =t4014, =t4012: y _ y - gcurtf.gtfch;
            =vcom, =singer, =comp80:
               BEGIN
               y _ y - gcurtf.gtfch - 14;
               yb _ yb - 14;
```

```
                END;
             ENDCASE;
          IF y<yb THEN EXIT LOOP;
          BUMP i;
          END;
       IF atvisible THEN
          gputat(atch, annotation, stid, 0, newflg);
       RETURN;
       END.
    (devset) PROCEDURE (gda);                                    6C1J6
       %Setup and control device status tables.  Caution - this
       routine may write on the device.%
       LOCAL scw, i;
       REF gda;
       %--------------------%
       gcurch _ fix(fmul(gda.gvtrn, float(gcurtf.gtfch)));
       gcurts_FALSE;
       CASE gdevtyp OF
          =t4012:
             IF (2*gcurch)<tekcsize[1] THEN gcurts_TRUE;
          =t4014:
             BEGIN
             %character size selection%
             IF (2*gcurch)<tekcsize[10] THEN gcurts_TRUE;
             i_2;
             %the values below are based on the table at
             tekcsize; however they are determined
             experimentally%
             IF gcurch IN[ 0, 40] THEN i_11;
             IF gcurch IN[41, 60] THEN i_8;
             IF gcurch IN[61, 88] THEN i_5;
             IF i#gtekch THEN
                BEGIN
                gpsend(tekesc);
                gpsend(tekcsize[i]);
                gtekch_i;
                END;
             %line style selection%
             i_teksty;
             IF gcurvc IN[1,5] THEN i_i+(gcurvc-1);
             IF gcurbd>5 THEN i_i+10B;  %defocus bit%
             IF i#gtekln THEN
                BEGIN
                gpsend(tekesc);
                gpsend(i);
                gtekln_i;
                END;
             RETURN;
             END;
          =vcom, =singer, =comp80:
             comch _ gcurch;
          ENDCASE;
       RETURN(TRUE);
       END.
    (gtline) PROCEDURE (gda, org, xl, xr, y, stng, fst, lst);
                                                                 6C1J7
```

```
%Output the text in stng[fst] to stng[lst] between xl
and xr on y in the page system using the viewing
transformation in the gda and the format data block
gcurtf. Returns false as soon as no further text in the
caption can be formated.%
LOCAL n, x, pcds, scds, pair[2], first, xl, yl, cw, ch,
wdth;
REF gda, stng;
%---------------%
n _ lst - fst + 1;
IF NOT n IN [1, 132] THEN RETURN (FALSE);
cw _ float(gcurtf.gtfcw);
ch _ float(gcurtf.gtfch);
wdth _ gcurtf.gtfcw*n;
CASE gcurtf.gtfjus OF
   =1:  %left justification%
      BEGIN
      x _ xl;
      END;
   =2:  %right justification%
      BEGIN
      x _ xr - (wdth);
      END;
   =3:  %center%
      BEGIN
      x _ xl  + ((xr - xl) - wdth)/2;
      END;
   ENDCASE  %full justification%
      BEGIN
      x _ xl;
      wdth _ xr - xl;
      END;
pcds.xc _ x;
pcds.yc _ y;
grptofp(pcds, $pair, org);
xl _ fadd(float(wdth), pair);
yl _ fadd(pair[1], ch);
%At this point pair is the floating point version of the
lower left cornor of the text and xl,yl is the upper
right cornor%
%If the line of text is above the window RETURN(TRUE)
because the next line of this caption may be visible.%
   IF pair[1] >= gda.gwtop THEN RETURN (TRUE);
   IF yl >= gda.gwtop THEN RETURN (TRUE);
%If the line of text is below the window RETURN(FALSE)
because the next line of this caption will never be
visible.%
   IF pair[1] < gda.gwbottom THEN RETURN (FALSE);
   IF yl <= gda.gwbottom THEN RETURN (FALSE);
%If the left edge of the text is to the right of the
window OR if the right edge of the text is to the left
of the window RETURN(TRUE).%
   IF pair >= gda.gwright THEN RETURN(TRUE);
   IF xl <= gda.gwleft THEN RETURN(TRUE);
%Now at least part of the string is in the window and
should be displayed.  The two branches below push the
```

```
        ends toward the window until both are visible.%
           UNTIL pair >= gda.gwleft DO
              BEGIN
              BUMP fst;
              pair _ fadd(pair, cw);
              END;
           UNTIL x1 <= gda.gwright DO
              BEGIN
              BUMP DOWN lst;
              x1 _ fsub(x1, cw);
              END;
        %The characters between fst and lst are visible - plot
        them.%
           CASE gdevtyp OF
              =t4012, =t4014: %Tektronix displays%
                 UNTIL fst > lst DO
                    BEGIN
                    scds _ gtrani(pair   , pair[1], &gda+gdavx);
                    gfatch(scds);
                    gpchar(scds, *stng*[fst]);
                    pair _ fadd(pair, cw);
                    BUMP fst;
                    END;
              =vcom, =singer, =comp80:  %Virtual com%
                 BEGIN
                 %Compute coordinates and counts%
                    n _ lst - fst + 1;
                    IF n<1 THEN EXIT CASE;
                 %Output a "Line Segment" - see journal - 14093%
                    gpsend(comnul);  gpsend(comnul);  %Who knows
                    why!%
                    gpsend(comlsg);
                    scds _ gtrani(pair   , pair[1], &gda+gdavx);
                    gccdsy(scds.yc);        %Base line%
                    gccdsx(scds.xc);        %Left edge%
                    scds _ gtrani(x1   , pair[1], &gda+gdavx);
                    gccdsx(scds.xc);        %Right edge%
                    gpsend(gcurtf.gtfjus-1);  %justification%
                    gpsend(1);              %one text segment
                    only%
                    gpsend(gcurtf.gtffnt);  %font%
                    gccdsx(comch); %character height%
                    gpsend(gcurtf.gtfemp);  %type style%
                    gccdsx(n);   %only one byte%
                    UNTIL fst>lst DO
                       BEGIN
                       gpsend(*stng*[fst]);
                       BUMP fst;
                       END;
                 END;
              ENDCASE;
        RETURN(TRUE);
        END.
     (gline) PROCEDURE (gda, x1, y1, x2, y2);                6C1J8
        %Display the line using the transformations (clip and
        view) in the gda.  Coordinates are floating.%
```

```
      LOCAL pair[4];
      LOCAL c1,c2;
      REF gda;
      pair    _ x1;
      pair[1] _ y1;
      pair[2] _ x2;
      pair[3] _ y2;
      IF gclip($pair, &gda+gdawdo) THEN
         %the coordinate pair is visible at least in part%
         BEGIN
         c1 _ gtrani(pair   ,pair[1], &gda+gdavx);
         c2 _ gtrani(pair[2],pair[3], &gda+gdavx);
         gfatch(c1);
         gfatch(c2);
         gppair(c1, c2);
         END;
      RETURN;
      END.
   (gclr) PROCEDURE;                                          6C1J9
      %clear tektronix screen%
      REF gprwndw;
      dfeclw(&gprwndw);  %do FE clear window%
      RETURN;
      END.
   (goutln) PROCEDURE (gda);                                  6C1J10
      %Outline the viewport for the given gda.%
      LOCAL double;
      REF gda;
      double _ 10;
      gpsopn();
      IF gda.gleft#scnleft THEN
         wtrect(gda.gleft+double, gda.gtop, gda.gleft-double,
         gda.gbottom);
      IF gda.gright#scnright THEN
         wtrect(gda.gright+double, gda.gtop,
         gda.gright-double, gda.gbottom);
      IF gda.gtop#scnvtop THEN
         wtrect(gda.gright, gda.gtop+double, gda.gleft,
         gda.gtop-double);
      IF gda.gbottom#scnbot THEN
         wtrect(gda.gright, gda.gbottom+double, gda.gleft,
         gda.gbottom-double);
      gpscls();
      RETURN;
      END.
   (goutsn) PROCEDURE;                                        6C1J11
      %Output the graphics static display.%
      LOCAL STRING s[35];
      gpsopn();
      wtrect(scnleft, scnbot, scnright, scntop);
      wtline(scnleft, scnvtop, scnright, scnvtop);
      *s* _ "NLS GRAPHICS";
      wtchar(scnleft+40, scnvtop+40, s.L, $s);
      gpscls();
      RETURN;
      END.
```

```
   (goutmgn) PROCEDURE (gda);                                    6C1J12
      %Display the margin settings.%
      LOCAL cds, pair[2];
      REF gda;
      IF NOT gmgnfl THEN RETURN;
      gpsopn();
      cds.xc _ gda.gmleft;
      cds.yc _ gda.gmtop;
      gtranf(cds, $pair, &gda+gdavx);
      cds _ gcftoi($pair);
      IF cds.xc IN [gda.gleft, gda.gright]
         THEN wtline(cds.xc, gda.gtop, cds.xc, gda.gbottom);
      IF cds.yc IN [gda.gbottom, gda.gtop]
         THEN wtline(gda.gleft, cds.yc, gda.gright, cds.yc);
      cds.xc _ gda.gmright;
      cds.yc _ gda.gmbottom;
      gtranf(cds, $pair, &gda+gdavx);
      cds _ gcftoi($pair);
      IF cds.xc IN [gda.gleft, gda.gright]
         THEN wtline(cds.xc, gda.gtop, cds.xc, gda.gbottom);
      IF cds.yc IN [gda.gbottom, gda.gtop]
         THEN wtline(gda.gleft, cds.yc, gda.gright, cds.yc);
      gpscls();
      RETURN;
      END.
%Graphics Data Area management%
   (galblk) PROCEDURE(blksiz);                                   6C1K1
      %Alocate a storage block and return its address%
      LOCAL gda;
      IF gda_getblk(blksiz, $dspblk) THEN
         RETURN(gda)
         ELSE RETURN(FALSE);
      END.
   (gfrblk) PROCEDURE(gda);                                      6C1K2
      %Free a storage block%
      IF NOT freeblk(gda, $dspblk) THEN
         err($"System error - gfrblk");
      RETURN(FALSE);
      END.
   (galstr) PROCEDURE(strsiz);  %Alocate a string from list
   zone%                                                         6C1K3
      LOCAL str;
      IF str_getstring(strsiz, lstzone) THEN
         RETURN(str)
         ELSE RETURN(FALSE);
      END.
   (gfrstr) PROCEDURE(str);                                      6C1K4
      %Free a string%
      IF NOT freestring(str, lstzone) THEN
         err($"System error - gfrstr");
      RETURN;
      END.
   (gallst) PROCEDURE(lstsize);  %Alocate a list%                6C1K5
      LOCAL list;
      IF list_getlst(lstsize) THEN
         RETURN(list)
```

```
            ELSE RETURN(FALSE);
         END.
      (gfrlst) PROCEDURE(list REF);  %Null and free storage for a
      list%                                                    6C1K6
         #list# _ ;
         IF NOT frelst(&list) THEN
            err($"System error - gfrlst");
         RETURN;
         END.
      (gcopda) PROCEDURE (gdadst, gdasrc);                     6C1K7
         %Copy gdasrc into gdadst%
         LOCAL i;
         REF gdadst, gdasrc;
         FOR i_0 UP UNTIL =gdalgt DO gdadst[i]_gdasrc[i];
         RETURN;
         END.
      (gcoput) PROCEDURE (gutdst, gutsrc);                     6C1K8
         %Copy gutsrc into gutdst%
         LOCAL i;
         REF gutdst, gutsrc;
         FOR i_0 UP UNTIL =(guntel * guntcl) DO
         gutdst[i]_gutsrc[i];
         RETURN;
         END.
   %Read specification file%
      (glodda) PROCEDURE(gdadst);                              6C1L1
         %Read the file given in the argument and load the
         gdadst%
         LOCAL i;
         REF gdadst;
         FOR i_0 UP UNTIL =gdalgt DO gdadst[i]_tstgda[i];
         gsdflt(&gdadst);  %default drawspecs%
         gdadst.gdares _ dfltr;  %default resolution%
         RETURN;
         END.
      (gsdflt) PROCEDURE (gda);                                6C1L2
         %Set drawspec defaults in gda%
         LOCAL TEXT POINTER ptr, ptr2;
         REF gda;
         %default character size, font, text window border and
         justification%
            FIND SF(*dfltcs*) ^ptr;
            gda.gdach _ gsetcs(&gda, $ptr: gda.gdacw);
            gda.gdafnt _ dfltcf;
            gda.gdabdr _ dfltb;
            gda.gdajus _ dfltj;
         %default arrow head size%
            FIND SF(*dfltah*) ^ptr SF(*dfltaw*) ^ptr2;
            gda.gdaahgt _ gsetas(&gda, $ptr, $ptr2: gda.gdaawdt);
         %default line type and style%
            gda.gdalvec _ dfltlt;
            gda.gdalint _ lstabl[(dfltls-1)*2];
            gda.gdalbrd _ lstabl[(dfltls-1)*2 + 1];
         %default grid%
            gda.gdagrd _ dfltg;
         RETURN;
```

```
      END.
   (glodut) PROCEDURE(gutdst);                                    6C1L3
      %Convert initial unit table to run-time format%
      LOCAL addr;
      LOCAL TEXT POINTER ptrf, ptre;
      REF gutdst, addr;
      &addr _ &gutdst;
      guntcl _ 0;
      FIND SF(*iunit*) ^ptrf;
      WHILE (FIND (ptrf > [",] ^ptre _ptre)) DO
         BEGIN
         IF guntcl > guntsz THEN
            err($"Too may unit definitions");
         addr.L _ 0;
         addr.M _ guntal;
         *addr* _ ptrf ptre;  %text string of unit
         abbreviation%
         FIND (ptre > CH ^ptrf);
         IF FIND (ptrf > [",] ^ptre _ptre) THEN
            addr.gutpsi _ gtxtfl ($ptrf,  $ptre)
         ELSE err($"Unit table format error");
         BUMP guntcl;
         &addr _ &addr + guntel;  %go to next entry%
         FIND (ptre > CH ^ptrf);
         END;
      RETURN;
      END.
%Graphics data structure manipulation%
   (gdgman) PROCEDURE (gda, mode, stid, dstid);                   6C1M1
      %Create, delete, copy and move diagrams.%
      LOCAL propdata[53], j;  %propdata must have size of
      "gdhsiz"%
      REF gda;
      CASE mode OF
         =gcreate:
            BEGIN
            %set up diagram header property%
            FOR j _ 0 UP UNTIL =gdhsiz DO
               propdata[j] _ 0;  %first zero out diagram
               header%
            propdata.gmargtop _ setmtop;
            propdata.gmargbottom _ setmbottom;
            propdata.gmargleft _ setmleft;
            propdata.gmargright _ setmright;
            propdata.gdhres _ dfltr;  %default resolution%
            IF NOT creprop(stid, dhtyp, gdhsiz, $propdata)
            THEN
               err($"unable to add diagram to statement");
            IF NOT gda.gtstid _ creit(stid, dhtyp) THEN
               BEGIN
               delprop(stid, dhtyp);
               err($"unable to add diagram to statement");
               END;
            gda.gdstid _ stid;
            gda.gcstid _ 0;
            gda.glstid _ 0;
```

```
                gda.gorg _ 0;
                gda.gmleft   _ setmleft;
                gda.gmright  _ setmright;
                gda.gmtop    _ setmtop;
                gda.gmbottom _ setmbottom;
                gda.gwleft   _ float(setmleft);
                gda.gwright  _ float(setmright);
                gda.gwtop    _ float(setmtop);
                gda.gwbottom _ float(setmbottom);
                gda.gdares _ dfltr;
                END;
             ENDCASE err(notyet);
          RETURN;
          END.
       (grpdhd) PROCEDURE (gda);                                    6C1M2
          %Reprop diagram header property block%
          LOCAL propdata[53], j;  %propdata must have size of
          "gdhsiz"%
          REF gda;
          %Set up full (new) size property block%
             FOR j _ 0 UP UNTIL =gdhsiz DO
                propdata[j] _ 0;
             propdata.gmargtop _ gda.gmtop;
             propdata.gmargbottom _ gda.gmbottom;
             propdata.gmargleft _ gda.gmleft;
             propdata.gmargright _ gda.gmright;
             propdata.gdhres _ gda.gdares;
          IF NOT reprop(gda.gdstid, dhtyp, gdhsiz, $propdata) THEN
             err ($"Bad file - grpdhd");
          RETURN;
          END.
       (gclman) PROCEDURE (gda, mode, type, tstid, dtstid);    6C1M3
          %Create, delete, copy and move cells.%
          LOCAL propdata[2], stid;
          REF gda;
          CASE mode OF
             =gcreate:
                BEGIN
                CASE type OF
                   =gpcell:
                      BEGIN
                      propdata _ 0;
                      propdata.gctype _ gpcell;
                      propdata.gcorg _ 0;
                      IF NOT gda.gcstid _ crenod(gda.gtstid, -1)
                      THEN err($"cannot create cell - gclman");
                      IF NOT creprop(gda.gcstid, chtyp, 2,
                      $propdata) THEN err($"cannot create cell -
                      gclman - 2");
                      IF NOT gda.glstid _ creit(gda.gcstid, chtyp)
                      THEN err($"cannot create cell - gclman -
                      3");
                      END;
                   ENDCASE err(notyet);
                END;
             ENDCASE err(notyet);
```

```
   RETURN;
   END.
(gckopn) PROCEDURE (gda);                                        6C1M4
   %Check the gda and insure a cell is open for writing.%
   LOCAL stid, stdb, blkadr;
   REF gda, blkadr;
   IF gda.gcstid THEN RETURN;
   %open the phantom cell if it exists%
   stid _ getsub(gda.gtstid);
   IF stid#gda.gtstid THEN
      IF lodprop(stid, chtyp : &blkadr, stdb) THEN
         IF blkadr[sdbhdl].gctype=gpcell THEN
            BEGIN
            gda.gcstid _ stid;
            gda.glstid _ getitree(stdb);
            RETURN;
            END;
   %create a phantom cell%
   gclman(&gda, gcreate, gpcell, gda.gtstid, 0);
   RETURN;
   END.
(gckemp) PROCEDURE (gda);                                        6C1M5
   %Check the gda to insure that a diagram is portrayed in
   viewport%
   REF gda;
   IF NOT gda.gdstid THEN
      err($"No diagram portrayed in viewport - gckemp");
   RETURN;
   END.
(gckgrp) PROCEDURE (scd1, scd2);                                 6C1M6
   %Check that scd1 and scd2 define a valid group%
   IF gdacds(scd1) # gdacds(scd2) THEN
      err($"Invalid group - gckgrp");
   RETURN;
   END.
(gbldgi) PROCEDURE (gda,type,subtype,rpcd1,rpcd2,rpcd3);
                                                                 6C1M7
   %Encode the data into a graphics instruction.  Store the
   result into the cell inferior tree at lstid.  Returns
   the address of the graphics instruction.  The usage of
   the rest of the argument list depends upon the figure
   type as follows:
      point -
      (gda, point, 0, relative page coordinates, 0, 0)
      line -
      (gda, line, 0, relative page coordinates, relative
      page coordinates, 0)
      rectangle -
      (gda, rectangle, 0, relative page coordinates of
      upper left cornor, width, depth, 0)
      diamond -
      (gda, diamond, 0, relative page coordinates of the
      left corner, half width, half height)
      circle -
      (gda, circle, 0, relative page coordinates of the
      center, radius, 0)
```

```
      arrowhead -
      (gda, arrowhead, direction code, relative page
      coordinates of the apex, half width, height)
      itriangle -
      (gda, itriangle, direction code, relative page
      coordinates of the apex, half width, height)
   %
   LOCAL gi, workarea, ptr, lstid;
   REF gi, gda, ptr;
   %---------------%
   lstid _ gda.glstid;
   CASE type OF
      =point:  %point%
         BEGIN
         &gi _ galogi(lstid, 2 : workarea);
         gfmtgi(&gi, &gda, type, 2);
         gi[1] _ rpcd1;     %end point%
         gputwa(workarea);
         END;
      =line:  %two point line%
         BEGIN
         &gi _ galogi(lstid, 3 : workarea);
         gfmtgi(&gi, &gda, type, 3);
         gi[1] _ rpcd1;     %end points%
         %additional points are relative to the first%
         gi[2].xc _ gsigne(rpcd2.xc) - gsigne(rpcd1.xc);
         gi[2].yc _ gsigne(rpcd2.yc) - gsigne(rpcd1.yc);
         gputwa(workarea);
         END;
      =rectangle:  %rectangle%
         BEGIN
         &gi _ galogi(lstid, 3 : workarea);
         gfmtgi(&gi, &gda, type, 3);
         gi[1] _ rpcd1;     %upper left cornor%
         gi[2].xc _ rpcd2;  %width%
         gi[2].yc _ rpcd3;  %height%
         gputwa(workarea);
         END;
      =diamond:  %diamond%
         BEGIN
         &gi _ galogi(lstid, 3 : workarea);
         gfmtgi(&gi, &gda, type, 3);
         gi[1] _ rpcd1;     %left cornor%
         gi[2].xc _ rpcd2;  %width%
         gi[2].yc _ rpcd3;  %height%
         gputwa(workarea);
         END;
      =itriangle:  %itriangle%
         BEGIN
         &gi _ galogi(lstid, 3 : workarea);
         gfmtgi(&gi, &gda, type, 3);
         gi[1] _ rpcd1;         %apex%
         gi[2].dir _ subtype;   %direction code%
         gi[2].wdt _ rpcd2;     %width%
         gi[2].hgt _ rpcd3;     %height%
         gputwa(workarea);
```

```
            END;
         =arrowhead:  %arrowhead%
            BEGIN
            &gi _ galogi(lstid, 3 : workarea);
            gfmtgi(&gi, &gda, type, 3);
            gi[1] _ rpcd1;          %apex%
            gi[2].dir _ subtype;  %direction code%
            gi[2].wdt _ rpcd2;    %width%
            gi[2].hgt _ rpcd3;    %height%
            gputwa(workarea);
            END;
         =circle:  %circle%
            BEGIN
            &gi _ galogi(lstid, 3 : workarea);
            gfmtgi(&gi, &gda, type, 3);
            gi[1] _ rpcd1;      %center%
            gi[2].yc _ rpcd2;  %radius%
            gi[2].xc _ 0;   %for the future%
            gputwa(workarea);
            END;
         ENDCASE err(notyet);
      &ptr _ gwaut[workarea].gwaaddr;
      RETURN(&gi, ptr.gwalstid, &gi-&ptr-gwahdl);
      END.
   (gfmtgi) PROCEDURE (gi, gda, type, length);                6C1M8
      %Build the first word of the gi.%
      REF gi, gda;
      %---------------%
      gi.gphops _ type;
      gi.gphlgt _ length;
      gi.gphmor _ FALSE;
      gi.gphatt _ FALSE;
      gi.gphvec _ gda.gdalvec;
      gi.gphint _ gda.gdalint;
      gi.gphbrd _ gda.gdalbrd;
      RETURN;
      END.
   (gbldtn) PROCEDURE (gda, rpcd1, width, height, sfnt, send);
                                                              6C1M9
      %Creat a text node at the end of the cell.  Return the
      stid of the node created.  Sfnt and send are text
      pointers to the beginning and end of the text to be
      captured.%
      LOCAL stid, propdata[5], i, lstid;
      REF sfnt, send, gda;
      %---------------%
      lstid _ gda.glstid;
      stid _ getail(getsub(lstid));
      stid _ cinssta(stid, 0, &sfnt, &send);
      %now create a format property on this node%
      propdata.gtfcds _ rpcd1;
      propdata.gtfwdt _ width;
      propdata.gtfhgt _ height;
      FOR i_2 UP UNTIL =5 DO propdata[i] _ gda[gdagtf+i-2];
      creprop(stid, gtftyp, 5, $propdata);
      RETURN (stid);
```

```
    END.
(gdelgi) PROCEDURE (stid,dsp,op);                                   6C1M10
    %The graphics instruction is removed from the file.%
    LOCAL size, i, j, workarea, ptr, ptr2, atadr, dsp2;
    REF ptr, ptr2, attach, atadr;
    %---------------%
    CASE op OF
        = annotation:
            BEGIN
            cdelgro(stid, stid, FALSE, 0);
            END;
        ENDCASE
            BEGIN
            %all graphics instructions processed here%
            workarea _ glodwa(stid);
            &ptr _ gwaut[workarea].gwaaddr + gwahdl + dsp;
            &ptr2 _ gwaut[workarea].gwaaddr;
            j _ ptr.gphlgt;
            mvbfbf(&ptr+j, &ptr, ((&ptr2 + gwahdl +
            ptr2.gwaclgt) - (&ptr + j)));
            ptr2.gwaclgt _ ptr2.gwaclgt - j;
            gputwa(workarea);
            %Update the attaching table if it exists%
            IF &attach THEN
                BEGIN
                &atadr _ &attach + gatfst;
                dsp2 _ &ptr - &ptr2 - gwahdl;
                FOR i_1 UP UNTIL >attach[gatcl] DO
                    BEGIN
                    IF atadr.gatstid=stid THEN
                        IF atadr.gatdsp>dsp2 THEN
                            BEGIN
                            atadr.gatdsp _ atadr.gatdsp - j;
                            END;
                    &atadr _ &atadr + gattel;
                    END;
                END;
            END;
    RETURN;
    END.
(gmovgi) PROCEDURE (stid,dsp,op,rpcd);                              6C1M11
    %Place the figure at rpcd.  Address of the updated gi in
    a workarea is returned for display.%
    LOCAL workarea, ptr, fst, blkadr, stdb, tfsav[5];
    REF ptr, blkadr;
    %---------------%
    CASE op OF
        = annotation:
            BEGIN
            lodprop(stid, gtftyp : &blkadr, stdb);
            mvbfbf (&blkadr+sdbhdl, $tfsav, 5);
            tfsav _ rpcd;
            reprop(stid, gtftyp, 5, $tfsav);
            RETURN (FALSE);
            END;
        ENDCASE
```

```
            BEGIN
            fst _ 1;  %use with the flag continuation logic%
            workarea _ glodwa(stid);
            &ptr _ gwaut[workarea].gwaaddr + gwahdl+ dsp;
            CASE ptr.gphops OF
               =line,  %line word graphics instruction%
               =point,  %point%
               =rectangle,  %rectangle%
               =diamond,  %diamond%
               =itriangle,  %itriangle%
               =arrowhead,  %arrowhead%
               =circle:  %circle%
                  BEGIN
                  ptr[fst] _ rpcd;
                  END;
               ENDCASE;
            gputwa(workarea);
            RETURN(&ptr);
            END;
      END.
(gcopgi) PROCEDURE (stid,dsp,op,rpcd,dstid);          6C1M12
      %A copy of the graphics instruction is placed in the
      cell addressed by dstid translated by rpcd.  Returns the
      address of the graphics instruction, stid, and
      displacement for linework  and the new stid for
      anotation.%
      LOCAL nstid, length, gi, workarea, fst, ptr, blkadr,
      stdb, page[512];
      REF gi, ptr, blkadr;
      %---------------%
      CASE op OF
         = annotation:
            BEGIN
            nstid _ ccopgro(dstid, 0, stid, stid, FALSE, 0);
            lodprop(nstid, gtftyp : &blkadr, stdb);
            mvbfbf (&blkadr+sdbhdl, $page, 5);
            page _ rpcd;
            reprop(nstid, gtftyp, 5, $page);
            RETURN(FALSE, nstid, 0);
            END;
         ENDCASE
            BEGIN
            fst _ 1;  %use with the flag continuation logic%
            IF NOT glodgi(stid, $page) THEN err($"node not
            found - gcopgi");
            length _ page[dsp].gphlgt;
            &gi _ galogi(dstid, length : workarea);
            mvbfbf($page[dsp], &gi, length);
            CASE gi.gphops OF
               =line,  %line word graphics instruction%
               =point,  %point%
               =rectangle,  %rectangle%
               =diamond,  %diamond%
               =itriangle,  %itriangle%
               =arrowhead,  %arrowhead%
               =circle:  %circle%
```

```
                    BEGIN
                    gi[fst] _ rpcd;
                    END;
                ENDCASE;
            gputwa(workarea);
            &ptr _ gwaut[workarea].gwaaddr;
            RETURN(&gi, ptr.gwalstid, &gi-&ptr-gwahdl);
            END;
    END.
(gexlgi) PROCEDURE (stid,dsp,extend,to,lstid);          6C1M13
    %A line gi is extended or contracted.  Returns the
    address of the graphics instruction, stid, and
    displacement.%
    LOCAL length, gi, workarea, fst, ptr, blkadr, stdb,
    page[512];
    REF gi, ptr, blkadr;
    %---------------%
    fst _ 1;  %use with the flag continuation logic%
    IF NOT glodgi(stid, $page) THEN err($"node not found -
    gcopgi");
    length _ page[dsp].gphlgt;
    CASE extend OF
        =20: %Extend visibly%
            BEGIN
            BUMP length;
            gdelgi(stid, dsp, line);
            &gi _ galogi(lstid, length : workarea);
            mvbfbf($page[dsp], &gi, length-1);
            gi.gphlgt _ length;
            gi[length-1].xc _ gsigne(to.xc) -
            gsigne(gi[fst].xc);
            gi[length-1].yc _ gsigne(to.yc) -
            gsigne(gi[fst].yc);
            gputwa(workarea);
            &ptr _ gwaut[workarea].gwaaddr;
            RETURN(&gi, ptr.gwalstid, &gi-&ptr-gwahdl);
            END;
        =22: %Extend invisibly%
            BEGIN
            length _ length + 2;
            gdelgi(stid, dsp, line);
            &gi _ galogi(lstid, length : workarea);
            mvbfbf($page[dsp], &gi, length-2);
            gi.gphlgt _ length;
            gi[length-2] _ gpenup;
            gi[length-1].xc _ gsigne(to.xc) -
            gsigne(gi[fst].xc);
            gi[length-1].yc _ gsigne(to.yc) -
            gsigne(gi[fst].yc);
            gputwa(workarea);
            &ptr _ gwaut[workarea].gwaaddr;
            RETURN(&gi, ptr.gwalstid, &gi-&ptr-gwahdl);
            END;
        =21: %Backspace%
            BEGIN
            IF length=fst+2 THEN err($"use delete line");
```

```
            gdelgi(stid, dsp, line);
            BUMP DOWN length;
            &gi _ galogi(lstid, length : workarea);
            mvbfbf($page[dsp], &gi, length);
            gi.gphlgt _ length;
            gputwa(workarea);
            &ptr _ gwaut[workarea].gwaaddr;
            RETURN(&gi, ptr.gwalstid, &gi-&ptr-gwahdl);
            END;
         ENDCASE;
      END.
   (glodgi) PROCEDURE (stid, page);                          6C1M14
      %Load a linework data block into the workarea at page.
      Returns size in words or False.%
      LOCAL blkadr, stdb, words;
      REF page, blkadr;
      IF NOT lodprop(stid, lwtyp : &blkadr, stdb) THEN RETURN
      (FALSE);
      words _ getdblen(stdb) - sdbhdl;
      mvbfbf(&blkadr+sdbhdl, &page, words);
      RETURN (words);
      END.
   (galogi) PROCEDURE (lstid,size);                          6C1M15
      %Space for a graphics instruction of length size is
      alocated within the cell lstid.  The work area workarea
      is used to store the line work property.  The core
      address of the free space is returned in the first
      argument while the index of the workarea alocated is
      returned in the second.%
      LOCAL ptr, gi, workarea, blkadr, stdb, stid, words;
      REF ptr, blkadr;
      %---------------%
      IF size>gwaml THEN err($"gi too big - galogi");
      IF NOT lstid THEN err($"system error - galogi");
      %look for sufficient workarea in core%
      FOR workarea_0 UP UNTIL =gwautl DO
         IF &ptr _ gwaut[workarea].gwaaddr THEN
            IF (ptr.gwacstid=lstid) AND
            (ptr.gwamlgt-ptr.gwaclgt > size) THEN
               BEGIN
               gi _ &ptr+gwahdl+ptr.gwaclgt;
               ptr.gwaclgt _ ptr.gwaclgt + size;
               RETURN(gi, workarea);
               END;
      %search the tree for a non-full linework property%
      stid _ getsub(lstid);
      LOOP
         BEGIN
         IF stid=lstid OR NOT lodprop(stid, lwtyp : &blkadr,
         stdb) THEN
            BEGIN
            %must alocate a new node%
            stid _ crenod(lstid, -1);
            workarea _ gbstwa(lstid);
            &ptr _ gwaut[workarea].gwaaddr;
            ptr.gwacstid _ lstid;
```

```
                ptr.gwalstid _ stid;
                ptr.gwamlgt _ gwaml;
                ptr.gwaclgt _ size;
                gi _ &ptr + gwahdl;
                creprop(stid, lwtyp, 0, 0);
                RETURN(gi, workarea);
                END;
             words _ getdblen(stdb) - sdbhdl;
             IF gwaml-words > size THEN
                BEGIN
                %move the data to the best workarea%
                workarea _ gbstwa(lstid);
                &ptr _ gwaut[workarea].gwaaddr;
                ptr.gwacstid _ lstid;
                ptr.gwalstid _ stid;
                ptr.gwamlgt _ gwaml;
                ptr.gwaclgt _ words + size;
                mvbfbf(&blkadr+sdbhdl, &ptr+gwahdl, words);
                gi _ &ptr+gwahdl+words;
                RETURN(gi, workarea);
                END;
             stid _ getsuc(stid);
             END;
          END.
    %Graphics display area manipulation%
       (gsetpv) PROCEDURE (gda);                                    6C1N1
          %The viewing and pointing transformations are created
          from the window and viewport dimensions and stored in
          the gda.  The window is expanded to fill the viewport
          while maintaining unity aspect ratio.%
          LOCAL wv, ww, hv, hw, fww, fhw;
          REF gda;
          %Setup and compute scale factors%
             wv _ float(gda.gright -gda.gleft);
             ww _ fsub(gda.gwright, gda.gwleft);
             hv _ float(gda.gtop -gda.gbottom);
             hw _ fsub(gda.gwtop, gda.gwbottom);
             gda.gvtrn _ gda.gvtrn5 _ MIN(fdiv(wv, ww), fdiv(hv,
             hw));
             gda.gptrn  _ fdiv(fone, gda.gvtrn);
             gda.gptrn5 _ fdiv(fone, gda.gvtrn5);
          %Fill up the viewport%
             fww _ fmul(gda.gptrn, wv);
             fhw _ fmul(gda.gptrn, hv);
             IF fww > ww THEN
                BEGIN
                fww _ fdiv(fsub(fww, ww), ftwo);
                gda.gwright _ fadd(gda.gwright, fww);
                gda.gwleft _ fsub(gda.gwleft, fww);
                END;
             IF fhw > hw THEN
                BEGIN
                fhw _ fdiv(fsub(fhw, hw), ftwo);
                gda.gwtop _ fadd(gda.gwtop, fhw);
                gda.gwbottom _ fsub(gda.gwbottom, fhw);
                END;
```

```
        %Viewing transformation%
           gda.gvtrn3 _ fsub(float(gda.gleft), fmul(gda.gvtrn,
           gda.gwleft));
           gda.gvtrn6 _ fsub(float(gda.gbottom), fmul(gda.gvtrn,
           gda.gwbottom));
           gda.gvtrn2 _ gda.gvtrn4 _ 0;
        %Point transformation%
           gda.gptrn3 _ fsub(0, fdiv(gda.gvtrn3, gda.gvtrn ));
           gda.gptrn6 _ fsub(0, fdiv(gda.gvtrn6, gda.gvtrn5));
           gda.gptrn2 _ gda.gptrn4 _ 0;
        RETURN;
        END.
     (glnkda) PROCEDURE (gda);                                  6C1N2
        %Link the gda to the list of gda's%
        REF gda, gdabas;
        gda.gdanxt_&gdabas;
        gda.gdapvs _ 0;
        gdabas.gdapvs_&gda;
        &gdabas_&gda;
        RETURN;
        END.
     (gdelda) PROCEDURE (gda);                                  6C1N3
        %Delete the gda from the list of gda's%
        LOCAL prvgda;
        REF gda, prvgda, gdabas;
        IF gda.gdapvs = 0
           THEN
              BEGIN
              &gdabas _ gda.gdanxt;  %deleting first gda in
              chain%
              gdabas.gdapvs _ 0;
              END
           ELSE
              BEGIN
              &prvgda _ gda.gdapvs;
              prvgda.gdanxt _ gda.gdanxt;
              END;
        gfrblk(&gda);
        RETURN;
        END.
  %FINISH.%%% % end of file 2 of support routines for split
```

```
version of subsystem%
%Graphics problem space routines%
   (gtxtpg) PROCEDURE (prmptr, gda);                              6C1P1
      %Returns number of page space units (in floating point)
      calculated from input text.%
      LOCAL unit, entry;
      LOCAL TEXT POINTER ptre, ptrf;
      REF prmptr, gda, entry, gunitb;
      IF FIND (SE(prmptr) ^ptre 1$(L) ^ptrf) THEN
         BEGIN  %letters at the end of the typein must be
         unit%
         IF NOT &entry _ gtxtut($ptrf, $ptre, &gunitb) THEN
         err($"Undefined unit abbreviation");
         FIND ptrf ^ptre;
         unit _ entry.gutpsi;
         END
      ELSE unit _ fone;   %Unit not entered.  assumes basic
      unit%
      RETURN (fmul(unit, fmul(gda.gdares, gtxtfl(&prmptr,
      $ptre))));
      END.
   (gtxtut) PROCEDURE (ptrf, ptre, table);                        6C1P2
      %Return address of matching entry in unit table or FALSE
      if no match. Input text is unit abbreviation.%
      LOCAL i, addr;
      REF ptrf, ptre, addr, table;
      IF ptre[1] - ptrf[1] > gtext.M THEN
         err($"Unit abbreviation too long");
      *gtext* _ -ptrf ptre;  %force to lower case%
      &addr _ &table;
      FOR i_ 0 UP UNTIL = guntcl DO  %search unit table%
         BEGIN
         IF *[&addr]* = *gtext* THEN RETURN(&addr);
         &addr _ &addr + guntel;  %go to next entry%
         END;
      RETURN(FALSE);
      END.
   (gtxtfl) PROCEDURE (ptrf, ptre);                               6C1P3
      %Return floating point value of number represented by
      input text %
      LOCAL number;
      REF ptrf, ptre;
      IF ptre[1] - ptrf[1] > gtext.M - 1 THEN
         err ($"Number too long");
      *gtext* _ ptrf ptre, "a;  %need "a to terminate number
      for flin%
      %Use JSYS to convert string to floating point number%
         R1 _ $gtext + chbmty;
         IF NOT SKIP !JSYS flin THEN
            err ($"Invalid number");
         number _ R2;
         RETURN (number);
      END.
   (gtxtin) PROCEDURE (ptrf, ptre);                               6C1P4
      %Return integer value of number represented by input
      text %
```

```
      LOCAL number;
      REF ptrf, ptre;
      IF ptre[1] - ptrf[1] > gtext.M - 1 THEN
         err ($"Number too long");
      *gtext* _ ptrf ptre, 'a;  %need 'a to teminate number
      for nin%
      %use JSYS to convert string to number%
         R1 _ $gtext + chbmty;
         R3 _ 10;
         IF NOT SKIP !JSYS nin THEN
            err ($"Invalid number");
         number _ R2;
         RETURN (number);
      END.
   (gfltxt) PROCEDURE (str, number);                          6C1P5
      %Add floating point number to end of string and update
      string length%
      LOCAL i, j, byteptr, saveptr;
      REF str;
      %Set byte pointer to 1 character after last character in
      string%
         byteptr _ &str + chbmty;
         FOR i _ 0 UP UNTIL = str.L DO
            j _ ^byteptr;
      saveptr _ byteptr;
      %Convert number to string using JSYS%
         R1 _ byteptr;
         R2 _ number;
         R3 _ 0;
         IF NOT SKIP !JSYS flout THEN
            err($"Invalid number in unit table - gfltxt");
         byteptr _ R1;
      %Update string length%
         DO
            BEGIN
            j _ ^saveptr;
            BUMP str.L;
            END
            UNTIL saveptr = byteptr;
      RETURN;
      END.
   (guedsp) PROCEDURE (str);                                  6C1P6
      %Add basic unit abbreviation and EOL to string and
      display  string%
      REF str;
      *str* _ *str*, " ", *[gbuabb]*;
      gedsp(&str);
      RETURN;
      END.
   (gedsp) PROCEDURE (str);                                   6C1P7
      %Add EOL to string and display%
      REF str;
      *str* _ *str*, EOL;
      gfbctl(fbaddlit, &str);
      RETURN;
      END.
```

```
%Work area management%
   (gbstwa) PROCEDURE (cstid);                                    6C1Q1
      %Return the index of the best workarea to use for the
      cell cstid.%
      LOCAL ptr, workarea, minage, minwa;
      REF ptr;
      %Look first for a workarea assigned to the cstid%
      FOR workarea_0 UP UNTIL =gwautl DO
         IF &ptr _ gwaut[workarea].gwaaddr THEN
            IF ptr.gwacstid=cstid THEN RETURN(workarea);
      %Look for an unalocated workarea%
      FOR workarea_0 UP UNTIL =gwautl DO
         IF NOT gwaut[workarea].gwaaddr THEN
            IF gwaut[workarea].gwaaddr _ galowa() THEN
            RETURN(workarea);
      %No new workarea are now possible use the youngest%
      minage _ 777777B;
      minwa _ 0;
      FOR workarea_0 UP UNTIL =gwautl DO
         IF gwaut[workarea].gwaage<minage THEN
            BEGIN
            minage _ gwaut[workarea].gwaage;
            minwa _ workarea;
            END;
         RETURN(minwa);
      END.
   (gputwa) PROCEDURE (workarea);                                 6C1Q2
      %The work area is returned to the file as a linework
      property block in the tree and node stored in the work
      area itself.%
      LOCAL ptr;
      REF ptr;
      &ptr _ gwaut[workarea].gwaaddr;
      IF ptr.gwalstid
         THEN reprop(ptr.gwalstid, lwtyp, ptr.gwaclgt,
         &ptr+gwahdl)
         ELSE err($"system error - gputwa");
      RETURN;
      END.
   (glodwa) PROCEDURE (stid);                                     6C1Q3
      %A linework property at stid is loaded in to a
      workarea.%
      LOCAL ptr, workarea, blkadr, stdb, words;
      REF ptr, blkadr;
      %See if the stid is already in a wa.%
      FOR workarea_0 UP UNTIL =gwautl DO
         IF &ptr _ gwaut[workarea].gwaaddr THEN
            IF ptr.gwacstid AND ptr.gwalstid=stid THEN RETURN
            (workarea);
      %Allocate one from lstid = 0%
      workarea _ gbstwa(0);
      IF NOT lodprop(stid, lwtyp : &blkadr, stdb) THEN
      err($"node not found - glodwa");
      &ptr _ gwaut[workarea].gwaaddr;
      words _ getdblen(stdb) - sdbhdl;
      ptr.gwacstid _ 0;
```

```
        ptr.gwalstid _ stid;
        ptr.gwaclgt _ words;
        ptr.gwamlgt _ gwaml;
        mvbfbf(&blkadr+sdbhdl, &ptr+gwahdl, words);
        RETURN (workarea);
        END.
    (galowa) PROCEDURE;                                          6C1Q4
        %Returns the address of a work area.%
        LOCAL workarea;
        IF workarea _ getblk(gwahdl+gwaml, $dspblk)
           THEN RETURN(workarea)
           ELSE RETURN(0);
        END.
    (gfrewa) PROCEDURE;                                          6C1Q5
        %All the workarea listed in the use table are
        eliminated.%
        LOCAL workarea;
        FOR workarea_0 UP UNTIL =gwautl DO
           BEGIN
           IF gwaut[workarea].gwaaddr THEN
              IF NOT freeblk(gwaut[workarea].gwaaddr, $dspblk)
              THEN err($"system error - gfrewa");
           gwaut[workarea] _ 0;
           END;
        RETURN;
        END.
%Pointing%
    (glda) PROCEDURE;                                            6C1R1
        %Returns the address of the gda for the last coordinate
        received and the last coordinate.%
        RETURN(gdacds(glastxy), glastxy);
        END.
    (gdacds) PROCEDURE (scds);                                   6C1R2
        %Returns the address of the gda for the coordinate.%
        LOCAL gda;
        REF gda, gdabas;
        &gda_&gdabas;
        IF NOT gda.gdanxt THEN RETURN (&gda);  %only one
        viewport%
        WHILE &gda DO
           BEGIN
           IF scds.xc IN[gda.gleft, gda.gright] AND
              scds.yc IN[gda.gbottom, gda.gtop] THEN RETURN
              (&gda);
           &gda_gda.gdanxt;
           END;
        &gda_&gdabas;
        WHILE &gda DO
           BEGIN
           IF scds.xc IN[gda.gleft, gda.gright] AND
           gda.gtop=scnvtop THEN RETURN (&gda);
           &gda_gda.gdanxt;
           END;
        err($"Viewport (ha ha) not found - gdacds");
        END.
    (gnghbr) PROCEDURE (gda, scds);                              6C1R3
```

```
    %Returns the address of the gda of the neighboring
    viewport to gda along the edge that contains the point
    scds%
    LOCAL gdangh;
    REF gda, gdabas, gdangh;
    &gdangh_&gdabas;
    WHILE &gdangh DO
       BEGIN
       IF scds.xc IN[gdangh.gleft, gdangh.gright] AND
          scds.yc IN[gdangh.gbottom, gdangh.gtop] AND
          &gdangh # &gda THEN RETURN (&gdangh);
       &gdangh_gdangh.gdanxt;
       END;
    err($"Please point to edge between 2 viewports");
    END.
(gdahv) PROCEDURE (gda, scds);                                  6C1R4
    %Determine which edge scds is closest to. returns 2
    values.
       value 1:  TRUE if closest to veritcal edge, else
       FALSE
       value 2:  top, bottom, left, right %
    LOCAL
       necont,  %near edge count%
       vebug,  %TRUE if vertical edge bugged%
       whiche;  %which edge%
    REF gda;
    necont _ 0;
    IF (scds.xc IN[gda.gleft-gatbox, gda.gleft+gatbox]) THEN
       BEGIN
       vebug _ TRUE;
       whiche _ ledge;
       BUMP necont;
       END;
    IF (scds.xc IN[gda.gright-gatbox, gda.gright+gatbox])
    THEN
       BEGIN
       vebug _ TRUE;
       whiche _ redge;
       BUMP necont;
       END;
    IF (scds.yc IN[gda.gbottom-gatbox, gda.gbottom+gatbox])
    THEN
       BEGIN
       vebug _ FALSE;
       whiche _ bedge;
       BUMP necont;
       END;
    %Use top edge if bug in teletype window and screen top
    is top edge of window%
    IF (scds.yc IN[gda.gtop-gatbox, gda.gtop+gatbox]) OR
       (scds.yc > gda.gtop AND gda.gtop =scnvtop) THEN
          BEGIN
          vebug _ FALSE;
          whiche _ tedge;
          BUMP necont;
          END;
```

```
      IF necont # 1 THEN
         err($"Please point to the edge more closely");
      RETURN(vebug, whiche);
      END.
   (gfind) PROCEDURE (opcode,scds);                          6C1R5
      %Scans to attaching table for a figure whose op code and
      attaching coordinates match.  Returns the address in the
      attaching table, gda, stid, and dsp for the figure if
      found, otherwise false.%
      LOCAL atadr, stid, gda, i;
      REF attach, atadr, gda;
      &atadr _ &attach + gatfst;
      FOR i_1 UP UNTIL >attach[gatcl] DO
         BEGIN
         IF atadr.gatop=opcode OR opcode=figure THEN
            IF gnear(scds, atadr.gatcd) THEN
               BEGIN
               &gda _ gdacds(scds);
               stid _ atadr.gatstid;
               RETURN(&atadr, &gda, stid, atadr.gatdsp);
               END;
         &atadr _ &atadr + gattel;
         END;
      RETURN (FALSE);
      END.
   (gnear) PROCEDURE (cd1, cd2);                             6C1R6
      %True if cd1 is "near" cd2.%
      IF (cd1.xc IN [cd2.xc-gatbox, cd2.xc+gatbox])
         AND
         (cd1.yc IN [cd2.yc-gatbox, cd2.yc+gatbox])
         THEN RETURN (TRUE);
      RETURN (FALSE);
      END.
   (ginside) PROCEDURE (cd1, wd1, wd2);                      6C1R7
      %True if cd1 is inside the window.%
      LOCAL x1, y1, x2, y2;
      x1 _ MIN(wd1.xc, wd2.xc);
      x2 _ MAX(wd1.xc, wd2.xc);
      y1 _ MAX(wd1.yc, wd2.yc);
      y2 _ MIN(wd1.yc, wd2.yc);
      IF (cd1.xc IN [x1, x2])
         AND
         (cd1.yc IN [y2, y1])
         THEN RETURN (TRUE);
      RETURN (FALSE);
      END.
   (gfatch) PROCEDURE (cds);                                 6C1R8
      %Update the attaching point%
      CASE cds.xc OF
         < atch.xc:
            BEGIN
            atvisible _ TRUE;
            atch.xc _ cds.xc;
            atch.yc _ cds.yc;
            END;
         = atch.xc:
```

```
                IF cds.yc > atch.yc THEN
                   BEGIN
                   atvisible _ TRUE;
                   atch.xc _ cds.xc;
                   atch.yc _ cds.yc;
                   END;
             ENDCASE;
          RETURN;
          END.
    %Attaching table manipulation%
       (gputat) PROCEDURE (scd,opcode,stid,dsp,newflg);        6C1S1
          %Store the data in the attaching table.%
          LOCAL w;
          REF attach,w;
          IF NOT &attach THEN RETURN;
          IF attach[gatcl]=attach[gatml] THEN
             gexpat(); %get more room%
          &w _ (gattel*attach[gatcl]);
          &w _ (&attach + gatfst + &w);
          w.gatcd _ scd;
          w.gatstid _ stid;
          w.gatdsp _ dsp;
          w.gatop _ opcode;
          w.gatnew _ newflg;
          BUMP attach[gatcl];
          RETURN;
          END.
       (gdelat) PROCEDURE (address,markflag);                  6C1S2
          %De-activate the attaching table entry at address.  If
          the markflag is true mark the display.%
          LOCAL cds;
          REF address, attach;
          cds _ address.gatcd;
          address.gatcd _ address.gatstid _ address.gatdsp _
          address.gatop _ 0;
          BUMP attach[gatgl];  %Increment delete count for garbage
          collect%
          IF markflag THEN
             BEGIN
             wtmark(cds.xc,cds.yc);
             END;
          RETURN;
          END.
       (giptrat) PROCEDURE;                                    6C1S3
          %Initialize next entry pointer to first entry%
          REF attach;
          attach[gatptr] _ gatfst;
          RETURN;
          END.
       (gnxtat) PROCEDURE;                                     6C1S4
          %Return address of next attach table entry or FALSE if
          at end of table%
          LOCAL nentry;
          REF attach;
          IF (attach[gatcl] = 0) OR (attach[gatptr] >= gatfst +
          gattel*attach[gatcl]) THEN
```

```
      nentry _ FALSE
   ELSE
      BEGIN
      nentry _ &attach + attach[gatptr];
      attach[gatptr] _ attach[gatptr] + gattel;
      END;
   RETURN(nentry);
   END.
(galoat) PROCEDURE (size);                                    6C1S5
   %Alocate an attaching table of size entries.%
   LOCAL newatt;
   REF attach, newatt;
   IF NOT &newatt _ getblk(gatfst+gattel*size,$dspblk) THEN
      BEGIN
      gpscls();  %fix up line processor%
      err($"Insufficient space for attaching table");
      END;
   &attach _ &newatt;
   attach[gatml] _ size;
   attach[gatcl] _ 0;
   attach[gatgl] _ 0;
   attach[gatptr] _ 0;
   RETURN;
   END.
(gclrat) PROCEDURE;                                           6C1S6
   %Clear the attaching table.%
   REF attach;
   IF NOT &attach THEN RETURN;
   attach[gatcl] _ 0;
   attach[gatgl] _ 0;
   RETURN;
   END.
(gexpat) PROCEDURE;                                           6C1S7
   %Provide more room for the attaching table.%
   LOCAL size,oldattach,i,j,n;
   REF attach,oldattach;
   IF attach[gatgl] > 0 THEN
      %Garbage collect the table.%
      BEGIN
      i _ j _ gatfst;
      WHILE j< gatfst + gattel*attach[gatcl]  DO
         BEGIN
         IF attach[gatptr] = j THEN
            attach[gatptr] _ i;  %New index to next entry%
         IF attach[j].gatop # 0 THEN
            BEGIN
            FOR n_0 UP UNTIL > gattel DO
               attach[i+n]_attach[j+n];
            i _ i + gattel;
            j _ j + gattel;
            END
         ELSE
            j _ j + gattel;
         END;
      attach[gatcl] _ attach[gatcl] - attach[gatgl];
      attach[gatgl] _ 0;
```

```
            END
         ELSE
            %Have to alocate more space from another block%
            BEGIN
            size _ attach[gatml] + gatexp;
            &oldattach _ &attach;
            galoat(size);
            attach[gatml] _ size;
            attach[gatcl] _ oldattach[gatcl];
            attach[gatgl] _ 0;
            attach[gatptr] _ oldattach[gatptr];
            FOR i_gatfst UP UNTIL >gatfst+gattel*size DO
               attach[i] _ oldattach[i];
            IF NOT freeblk(&oldattach,$dspblk) THEN
               err($"System error - gexpat");
            END;
         RETURN;
         END.
      (gfreat) PROCEDURE;                                        6C1S8
         %Eliminate the attaching table.%
         REF attach;
         IF NOT &attach THEN RETURN;
         IF NOT freeblk(&attach,$dspblk) THEN
            err($"System error - gfreat");
         &attach _ 0;
         RETURN;
         END.
   %Coordinate transformations%
      (gridxy) PROCEDURE (gda,x,y);                              6C1T1
         %The point (x,y) is converted from screen coordinates to
         page coordinates and constrained to the current grid.%
         LOCAL cds, pair[2];
         REF gda;
         %convert x and y to a single word%
            cds _ gpack (x, y);
         %convert to page coordinates%
            gtranf(cds,$pair,&gda+gdapx);
         %convert the floating point result to integer%
            cds _ gcftoi($pair);
         %resolve the coordinates to the current grid%
            cds.xc _ greslv(&gda, cds.xc);
            cds.yc _ greslv(&gda, cds.yc);
         RETURN (cds);
         END.
      (greslv) PROCEDURE (gda, coord);                           6C1T2
         %Round the coordinate to the grid given in the gda.%
         LOCAL ngrid, fullwd, sign;
         REF gda;
         sign _ gspstv(gsigne(coord): fullwd);
         %Calculate integral number of grid increments%
            ngrid _ grfix(fdiv(float(fullwd), gda.gdagrd));
         %Calculate closest grid point%
            RETURN (grfix(fmul(float(ngrid), gda.gdagrd)) *
            sign);
         END.
      (gngrid) PROCEDURE (coord, gda);                           6C1T3
```

```
      %Return grid increment for the coordinate %
      LOCAL fullwd, sign;
      REF gda;
      sign _ gspstv(gsigne(coord): fullwd);
      RETURN(grfix(fdiv(float(fullwd), gda.gdagrd)) * sign);
      END.
   (gcdgrd) PROCEDURE (gincrm, gda);                          6C1T4
      %Return coordinate for grid increment%
      LOCAL sign, postiv;
      REF gda;
      sign _ gspstv(gincrm: postiv);
      RETURN (grfix(fmul(float(postiv), gda.gdagrd)) * sign);
      END.
   (gspstv) PROCEDURE (fullwd);                               6C1T5
      %Return sign (1 if fullwd is positive, -1 if fullwd is
      negative) and absolute value of fullwd%
      IF fullwd >= 0 THEN
         RETURN(1, fullwd)
      ELSE RETURN(-1, -fullwd);
      END.
   (gridnc) PROCEDURE (sgda, dgda, atadr, from, to);          6C1T6
      %Compute the new value of the origin of the graphics
      instruction from the screen coordinate data passed.%
      LOCAL gicds, orgcds, orgch, frmcds, tocds, blkadr, stdb,
      fst;
      REF sgda, dgda, atadr, blkadr;
      %find the origin of the source cell.  get cell header by
      using pointer to sdb in origin ofof line work rings%
         IF NOT (stdb _ getphed(atadr.gatstid)) THEN
            err($"Inferior tree format error - gridnc");
         lodent(stdb, sdbtyp : &blkadr);
         orgch _ blkadr[sdbhdl].gcorg;
      %find the orgin of the graphics instruction or text
      format block%
         fst _ 1;  %point to the origin of the graphics
         instruction%
         IF atadr.gatop=annotation THEN
            BEGIN
            IF NOT lodprop(atadr.gatstid, gtftyp : &blkadr,
            stdb) THEN
               err($"node not found - gridnc");
            orgcds _ blkadr[sdbhdl].gtfcds;
            END
         ELSE
            BEGIN
            IF NOT lodprop(atadr.gatstid, lwtyp : &blkadr,
            stdb) THEN
               err($"node not found - gridnc");
            orgcds _ blkadr[sdbhdl  + atadr.gatdsp + fst];
            END;
      %convert the coordinates to the page space%
         frmcds _ gridxy(&sgda, from.xc, from.yc);
         tocds _ gridxy(&dgda, to.xc, to.yc);
      %use half word arithmetic to compute new coordinate%
         frmcds _ ghsub(frmcds, orgch);
         tocds _ ghsub(tocds, dgda.gorg);
```

```
      gicds _ ghadd(tocds, ghsub(orgcds, frmcds));
   RETURN (gicds);
   END.
(gtranf) PROCEDURE (coords,cresult,trans);                    6C1T7
   %transform the integer coordinate pair in coords by the
   transformation addressed by trans.  The floating result
   is stored at cresult (2 words).%
   REF cresult;
   gcitof(coords,&cresult);
   %unity transformation exits directly%
   IF trans=0 THEN RETURN;
   R0 _ cresult;
   R1 _ cresult[1];
   R2 _ trans;
   !MOVE R3,R0;      %need two copies of x and y%
   !MOVE R4,R1;
   !FMPR R0,0(R2);   %develope ax,by,dx,ey%
   !FMPR R1,1(R2);
   !FMPR R3,3(R2);
   !FMPR R4,4(R2);
   !FADR R0,R1;      %sum x' _ ax + by + c, y' _ dx + ey +
   f%
   !FADR R0,2(R2);
   !FADR R3,R4;
   !FADR R3,5(R2);
   cresult _ R0;
   cresult[1] _ R3;
   RETURN;
   END.
(gtrani) PROCEDURE (x,y,trans);                               6C1T8
   %The floating point coordinates stored in x and y are
   transformed by the  transformation addressed by trans.
   The packed integer coordinate pair is returned.%
   R0 _ x;
   R1 _ y;
   R3 _ y;
   %unity transformation skips the multiplication%
   IF trans#0 THEN
      BEGIN
      R2 _ trans;
      !MOVE R3,R0;      %need two copies of x and y%
      !MOVE R4,R1;
      !FMPR R0,0(R2);   %develope ax,by,dx,ey%
      !FMPR R1,1(R2);
      !FMPR R3,3(R2);
      !FMPR R4,4(R2);
      !FADR R0,R1;      %sum x' _ ax + by + c, y' _ dx + ey
      + f%
      !FADR R0,2(R2);
      !FADR R3,R4;
      !FADR R3,5(R2);
      END;
   !MOVE R4,R0;      %fix the result%
   !MULI R0,400B;
   !EXCH R1;
   !ASH R0,-243B(R1);
```

```
      !SKIPGE R4;
      !MOVNS R0;
      !MOVE R1,R3;
      !MULI R1,400B;
      !EXCH R2,R1;
      !ASH R1,-243B(R2);
      !SKIPGE R3;
      !MOVNS R1;
      %***** note that x and y are stored without byte
      pointers xc and yc *****%
      R0.LH _ R1;
      RETURN(R0);
      END.
   (ghadd) PROCEDURE (cd1, cd2);                              6C1T9
      %Add sign extended halfwords in cd1 to cd2.%
      LOCAL cds;
      cds.xc _ gsigne(cd1.xc) + gsigne(cd2.xc);
      cds.yc _ gsigne(cd1.yc) + gsigne(cd2.yc);
      RETURN (cds);
      END.
   (ghsub) PROCEDURE (cd1,cd2);                               6C1T10
      %Subtract sign extended halfwords in cd2 from cd1.%
      LOCAL cds;
      cds.xc _ gsigne(cd1.xc) - gsigne(cd2.xc);
      cds.yc _ gsigne(cd1.yc) - gsigne(cd2.yc);
      RETURN(cds);
      END.
   (gpack) PROCEDURE (x,y);                                   6C1T11
      %Convert x,y to a single packed word%
      LOCAL cds;
      cds.xc _ x;
      cds.yc _ y;
      RETURN(cds);
      END.
   (gsigne) PROCEDURE (cd);                                   6C1T12
      %Sign extend an 18 bit coordinate.%
      !HRRES cd;
      RETURN(cd);
      END.
   (gcitof) PROCEDURE (cds,pair);                             6C1T13
      %The integer coordinate pair cds is converted to a
      floating point coordinate pair (x,y) and stored at
      pair.%
      REF pair;
      pair   _float(gsigne(cds.xc));
      pair[1]_float(gsigne(cds.yc));
      RETURN;
      END.
   (gcftoi) PROCEDURE (pair);                                 6C1T14
      %The floating point coordinate pair (x,y) is converted
      to a single word Integer coordinate.%
      LOCAL cds;
      REF pair;
      cds.xc _ fix(pair);
      cds.yc _ fix(pair[1]);
      RETURN(cds);
```

```
        END.
      (grptofp) PROCEDURE (cds, pair, org);                    6C1T15
        %Relative page to floating point page conversion.%
        LOCAL xorg, yorg, x, y;
        REF pair;
        xorg _ org.xc;
        yorg _ org.yc;
        x _ cds.xc;
        y _ cds.yc;
        %***** half word sign extension used *****%
        !HRRES xorg;
        !HRRES yorg;
        !HRRES x;
        !HRRES y;
        pair    _ float(x+xorg);
        pair[1] _ float(y+yorg);
        RETURN;
        END.
    %Clipping%
      (gclip) PROCEDURE (pair,window);                         6C1U1
        %clip the coordinate pair addressed by
        pair(=x1,y1,x2,y2) to the window addressed by
        window(=xl,xr,ytop,ybottom)%
        LOCAL c1,c2;
        REF pair,window;
        %---------------%
        c1 _ gcode(pair   ,pair[1],&window);
        c2 _ gcode(pair[2],pair[3],&window);
        WHILE (c1+c2) DO
          BEGIN
          IF c1 .A c2 THEN RETURN(FALSE); %wholly invisible%
          IF c1=0 THEN
            BEGIN
            c1 _ c2 := c1;  %swap point 1 and 2 to insure c1
            out%
            pair    _ pair[2] := pair;
            pair[1] _ pair[3] := pair[1];
            END;
          %compute the intersection of the line with a
          boundary%
          IF (c1 .A 1) THEN
            BEGIN
            R0 _ pair[3];
            R3 _ pair;
            R4 _ pair[1];
            !FSBR R0,R4;
            R1 _ window;
            !FSBR R1,R3;
            R2 _ pair[2];
            !FSBR R2,R3;
            !FMPR R0,R1;
            !FDVR R0,R2;
            !FADR R0,R4;
            pair[1] _ R0;
            pair _ window;
            END
```

```
            ELSE IF (c1 .A 2) THEN
               BEGIN
               R0 _ pair[3];
               R3 _ pair;
               R4 _ pair[1];
               !FSBR R0,R4;
               R1 _ window[1];
               !FSBR R1,R3;
               R2 _ pair[2];
               !FSBR R2,R3;
               !FMPR R0,R1;
               !FDVR R0,R2;
               !FADR R0,R4;
               pair[1] _ R0;
               pair _ window[1];
               END
            ELSE IF (c1 .A 4) THEN
               BEGIN
               R0 _ pair[2];
               R3 _ pair;
               R4 _ pair[1];
               !FSBR R0,R3;
               R1 _ window[3];
               !FSBR R1,R4;
               R2 _ pair[3];
               !FSBR R2,R4;
               !FMPR R0,R1;
               !FDVR R0,R2;
               !FADR R0,R3;
               pair _ R0;
               pair[1] _ window[3];
               END
            ELSE IF (c1 .A 10B) THEN
               BEGIN
               R0 _ pair[2];
               R3 _ pair;
               R4 _ pair[1];
               !FSBR R0,R3;
               R1 _ window[2];
               !FSBR R1,R4;
               R2 _ pair[3];
               !FSBR R2,R4;
               !FMPR R0,R1;
               !FDVR R0,R2;
               !FADR R0,R3;
               pair _ R0;
               pair[1] _ window[2];
               END;
            c1 _ gcode(pair,pair[1],&window);
            END;
         RETURN(TRUE);  %the pair is visible%
         END.
      (gcode) PROCEDURE (x,y,window);                                  6C1U2
         %compute the 'outcode' for the point x,y for the window
         addressed by window%
         %*****  caution x,y and window are single precision
```

```
        floating point numbers *****%
        REF window;
        RETURN
           ((IF x<window THEN 1 ELSE
           IF x>window[1] THEN 2 ELSE 0) +
           (IF y<window[3] THEN 4 ELSE
           IF y>window[2] THEN 10B ELSE 0));
        END.
     %Floating point math%
        (grfix) PROCEDURE (fparg);                                  6C1V1
           %The result is an integer whoe value is the floating
           point argument rounded to the nearest integer%
           LOCAL fixnmb;
           fixnmb _ fix(fparg);
           RETURN
              (IF fsub(fparg, float(fixnmb)) > fhalf THEN
                 fixnmb + 1
              ELSE fixnmb );
           END.
        (fadd) PROCEDURE (arg1,arg2);                               6C1V2
           %Result is the floating point arg1 + arg2.%
           R0 _ arg1;
           R1 _ arg2;
           !FADR R0,R1;
           RETURN (R0);
           END.
        (fsub) PROCEDURE (arg1,arg2);                               6C1V3
           %Result is the floating point arg1 - arg2.%
           R0 _ arg1;
           R1 _ arg2;
           !FSBR R0,R1;
           RETURN (R0);
           END.
        (fmul) PROCEDURE (arg1,arg2);                               6C1V4
           %Result is the floating point arg1 * arg2.%
           R0 _ arg1;
           R1 _ arg2;
           !FMPR R0,R1;
           RETURN (R0);
           END.
        (fdiv) PROCEDURE (arg1,arg2);                               6C1V5
           %Result is the floating point arg1 / arg2.%
           R0 _ arg1;
           R1 _ arg2;
           !FDVR R0,R1;
           RETURN (R0);
           END.
        (fix) PROCEDURE (fparg);                                    6C1V6
           %The result is the integer part of the floating point
           argument.%
           R2 _ fparg;
           !MOVM R0,R2;
           !MULI R0,400B;
           !EXCH R0,R1;
           !ASH R0,-243B(R1);
           !SKIPGE R2;
```

```
    !MOVNS R0;
    RETURN (R0);
    END.
(float) PROCEDURE (iarg);                                    6C1V7
    %The result is the floating point representation of the
    integer argument.%
    R2 _ iarg;
    !MOVM R0,R2;
    !HLRZ R1,R0;
    !FSC R1,255B;
    !HRLI R0,0;
    !FSC R0,233B;
    !FADR R0,R1;
    !SKIPGE R2;
    !MOVN R0,R0;
    RETURN (R0);
    END.
(fdrad) PROCEDURE (angle);                                   6C1V8
    %Convert the angle (degrees) to radians.%
    R0 _ angle;
    R1 _ 1734357506508;  %0.0174532925%
    !FMPR R0,R1;
    RETURN (R0);
    END.
(fradd) PROCEDURE (angle);                                   6C1V9
    %Convert the angle (radians) to degrees.%
    R0 _ angle;
    R1 _ 2067122734078;  %57.2957795786%
    !FMPR R0,R1;
    RETURN (R0);
    END.
(fsin) PROCEDURE (angle);                                   6C1V10
    %The floating point angle in radians is converted to its
    sine.%
    RETURN (fsico(angle)); END.
(fcos) PROCEDURE (angle);                                   6C1V11
    %The angle is converted to its cosine.%
    R0 _ angle;
    R1 _ 2016220773258;  %1.5707963%
    !FADR R0,R1;
    RETURN (fsico(R0)); END.
(fsico) PROCEDURE (angle);                                  6C1V12
    %Common part of sine  and cosine.  (code transliterated
    from the Fortran runtime system)%
    R1 _ angle;
    R6 _ R1;
    !MOVMS R1;
    IF R1 > 170000000000B THEN
       BEGIN
       R4 _ 2016220773258; %1.5707963%
       !FDV R1,R4;
       IF R1 > 201400000000B %1.0% THEN
          BEGIN
          !MULI R1,400B;
          !LSH R2,-202B(R1);
          !TLZ R2,400000B;
```

```
                !MOVEI R1,200B;
                !ROT R2,R3;
                !LSHC R1,33B;
                R5 _ 0;
                !FAD R1,R5;  %this must be a normalization trick%
                IF R2 # 0 THEN
                   BEGIN
                   R5 _ 2014000000000B; %1.0%
                   !TLCE R2,1000B;
                   !FSB R1,R5;
                   !TLCE R2,3000B;
                   !TLNN R2,3000B;
                   !MOVNS R1;
                   END;
                END;
             !SKIPGE R6;
             !MOVNS R1;
             R6 _ R1;
             !FMPR R1,R1;
             R0 _ 1644755367228; %0.15148418E-3%
             !FMP R0,R1;
             R5 _ 6063155463468; %-0.46737656E-2%
             !FAD R0,R5;
             !FMP R0,R1;
             R5 _ 1755063212768; %0.79689680E-1%
             !FAD R0,R5;
             !FMP R0,R1;
             R5 _ 5772652103728; %-0.64596371%
             !FAD R0,R5;
             !FMP R0,R1;
             !FAD R0,R4;
             !FMPR R0,R6;
             RETURN (R0);
             END
          ELSE
             RETURN (R6);
             END.
       (fsqrt) PROCEDURE (val);                                    6C1V13
          %Obtain the square root of val.  Returns 0 if less or
          equal 0.%
          IF val <= 0 THEN RETURN(0);
          R1 _ val;
          R0 _ 0;
          !LSHC R0,11B;
          !SUBI R0,201B;
          !ROT R0,-1;
          !HRRM R0,R4;
          !LSH R1,-11B;
          IF R0 < 0 THEN
             BEGIN
             !FSC R1,200B;
             R3 _ R1;
             !FMPRI R1,200450B;
             !FADRI R1,177660B;
             END
          ELSE
```

```
            BEGIN
            !FSC R1,177B;
            R3 _ R1;
            !FMPRI R1,200640B;
            !FADRI R1,1774658;
            END;
         R0 _ R3;
         !FDV R0,R1;
         !FAD R1,R0;
         !FSC R1,-1;
         R0 _ R3;
         !FDV R0,R1;
         !FADR R0,R1;
         !FSC R0,(R4);
         RETURN(R0);
         END.
      (iabs) PROCEDURE (n);                                    6C1V14
         %return the absolute value%
         RETURN(IF n>0 THEN n ELSE -n);
         END.
      (fabs) PROCEDURE (n);                                    6C1V15
         %return the absolute value of a floating point number%
         RETURN(IF n>0 THEN n ELSE fsub(0, n));
         END.
      (absdif) PROCEDURE (h1, h2);                             6C1V16
         %return the absolute value of the difference between two
         half-words%
         RETURN ( iabs(gsigne(h1) - gsigne(h2)) );
         END.
      (maxse) PROCEDURE (h1, h2);                              6C1V17
         %return the maximum half-word%
         RETURN ( MAX(gsigne(h1), gsigne(h2)) );
         END.
      (minse) PROCEDURE (h1, h2);                              6C1V18
         %return the minimum half-word%
         RETURN ( MIN(gsigne(h1), gsigne(h2)) );
         END.
   %Graphics protocol AND buffering%
      (g4cdst) PROCEDURE (x,y);                                 6C1W1
         %Output the four coordinate characters to the
         Tektronix.%
         x _ x+gxblrr; %x blurring%
         y _ y+gyblrr; %y blurring%
         gpsend(hybyte+y.hi);
         IF gdevtyp=t4014 THEN gpsend(lybyte+4*y.xb+x.xb);
         gpsend(lybyte+y.lo);
         gpsend(hxbyte+x.hi);
         gpsend(lxbyte+x.lo);
         RETURN;
         END.
      (gccdsx) PROCEDURE (cd);                                  6C1W2
         %Output the 2 byte x coordinate to the com files only%
         gpsend(cd.b7high);
         gpsend(cd.b7low);
         RETURN;
         END.
```

```
(gccdsy) PROCEDURE (cd);                                        6C1W3
   %Output the 2 byte y coordinate to the com files only%
   LOCAL cds;
   cds _ comybfb - cd;
   gpsend(cds.b7high);
   gpsend(cds.b7low);
   RETURN;
   END.
(gpsopn) PROCEDURE;  %Initialize output for the specific
device%                                                         6C1W4
   REF sendbf, gcommands;
   CASE gdevtyp OF
      =t4012, =t4014:
         BEGIN
         % Set up graphics instructions buffer %
            &sendbf _ galstr(gbfsize);  %alloc. string%
            sendp _ chbmty + &sendbf;  %initialize byte
            pointer%
            sendl _ 0;  %initialize character count%
         tekmode _ alpham;
         %start commands list to Fe with turn tracking off
         command%
            #gcommands# _ ;  %null out list%
            bldtrk(FALSE, &gcommands);
         END;
      =vcom, =singer, =comp80:  %virtual COM device%
         NULL;
      ENDCASE;
   RETURN;
   END.
(gpsend) PROCEDURE (char);  %Process a single character%
                                                                6C1W5
   REF sendbf, gcommands, gprwndw,
      adout1;  %address of the byte output routine%
   CASE gdevtyp OF
      =t4012, =t4014:
         BEGIN  %This is lineprocessor protocol%
         ^sendp _ char;
         BUMP sendl;
         IF sendl = gbfsize THEN
            BEGIN  %string is full.  send to FE%
            sendbf.L _ gbfsize;
            %build FE write literal%
               bldwli(&gprwndw, &sendbf, &gcommands);
            % Set up graphics instructions buffer %
               &sendbf _ galstr(gbfsize);  %alloc. string%
               sendp _ chbmty + &sendbf;  %initialize byte
               pointer%
               sendl _ 0;  %initialize character count%
            IF gcommands.L = maxfei THEN
               BEGIN  %this will fill message buffer%
               prcmds(0, &gcommands, 0, 0);  %send commands
               to FE%
               #gcommands# _ ;  %null out commands list%
               END;
            END;
```

```
            END;
         =vcom:  %virtual COM device%
            !bout(comjfn, char);
         =singer, =comp80:  %Output prrocessor%
            adout1(char);
         ENDCASE;
      RETURN;
      END.
   (gpscls) PROCEDURE;                                          6C1W6
      %Terminate output for the specific device%
      REF sendb, gcommands, gprwndw;
      CASE gdevtyp OF
         =t4012, =t4014:
            BEGIN
            IF tekmode=graphm THEN
               BEGIN
               gpsend(tekpad);
               gpsend(tekpad);
               gpsend(tekpad);
               gpsend(tekam);
               tekmode _ alpham;
               END;
            IF sendl THEN
               BEGIN  %Send data that is currently in buffer%
               sendbf.L _ sendl;
               bldwli(&gprwndw, &sendbf, &gcommands);
               END;
            bldtrk(TRUE, &gcommands);  %command to turn
            tracking on%
            prcmds(0, &gcommands, 0, 0);  %send commands to
            FE%
            #gcommands# _ ;  %null out commands list%
            END;
         =vcom, =singer, =comp80:  %virtual COM device%
            NULL;
         ENDCASE;
      RETURN;
      END.
   (gpchar) PROCEDURE (c1, char);                               6C1W7
      %Output character char at location c1 using the format
      information in the display device state table.%
      CASE gdevtyp OF
         =t4012, =t4014:
            BEGIN
            gpsend(tekgm);
            g4cdst(c1.xc, c1.yc);
            gpsend(tekam);
            IF gcurts THEN gpsend("`) ELSE gpsend(char);
            tekmode _ alpham;
            END;
         =vcom, =singer, =comp80:  %virtual COM device%
            BEGIN
            %Output a "Line Segment"%
            gpsend(comnul);  gpsend(comnul);  %Who knows why!%
            gpsend(comlsg); %see journal - 14093%
            gccdsy(c1.yc);
```

```
                gccdsx(c1.xc);
                gccdsx(c1.xc);  %??%
                gpsend(0);  %flush left - graphics system
                justifies%
                gpsend(1);  %one byte only%
                gpsend(gcurtf.gtffnt);  %font%
                gccdsx(comch); %interline spacing%
                gpsend(comtyp);  %type style%
                gccdsx(1);   %only one byte%
                gpsend(char);  %the data%
                END;
             ENDCASE;
          RETURN;
          END.
       (gppair) PROCEDURE (c1,c2);                                    6C1W8
          %Output a line between the two coordinates c1 and c2.%
          CASE gdevtyp OF
             =t4012, =t4014:
                BEGIN
                IF c2=clast THEN c1 _ c2 := c1; %be sure c1 is
                clast if possible%
                IF c1#clast THEN
                   %must start a new line%
                   BEGIN
                   gpsend(tekgm);
                   tekmode _ graphm;
                   g4cdst(c1.xc,c1.yc);  %blank line to one end%
                   END;
                %continue the line%
                g4cdst(c2.xc,c2.yc);
                clast _ c2;
                END;
             =vcom, =singer, =comp80:  %virtual COM device%
                BEGIN
                %Output a line%
                gpsend(comnul);  gpsend(comnul);  %Who knows why!%
                gpsend(comln);   %linework op code%
                gpsend(gcurvc);  %line type%
                gpsend(gcurin);  %intensity%
                gpsend(gcurbd);  %broadness%
                gccdsx(c1.xc);
                gccdsy(c1.yc);
                gccdsx(c2.xc);
                gccdsy(c2.yc);
                END;
             ENDCASE;
          RETURN;
          END.
       %Direct tektronix output%
          (wtmark) PROCEDURE (x1,y1);                                 6C1X1
             %draw an "X" mark at the pint x1,y1%
             LOCAL size;
             size _ 40;
             gpsend(tekgm);
             g4cdst(x1-size,y1-size);
             g4cdst(x1+size,y1+size);
```

```
    gpsend(tekgm);
    g4cdst(x1+size,y1-size);
    g4cdst(x1-size,y1+size);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekam);
    RETURN ;
    END.
(wtrect) PROCEDURE (x1,y1,x2,y2);                          6C1X2
    %draw a rectangle%
    gpsend(tekgm);
    g4cdst(x1,y1);
    g4cdst(x2,y1);
    g4cdst(x2,y2);
    g4cdst(x1,y2);
    g4cdst(x1,y1);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekam);
    RETURN ;
    END.
(wtline) PROCEDURE (x1,y1,x2,y2);                          6C1X3
    %output a line to the tektronix%
    gpsend(tekgm);
    g4cdst(x1,y1);
    g4cdst(x2,y2);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekam);
    RETURN;
    END.
(wtpnt) PROCEDURE (x1,y1);                                 6C1X4
    %output a point to the tektronix%
    gpsend(tekgm);
    g4cdst(x1,y1);
    g4cdst(x1,y1);
    gpsend(tekam);
    RETURN;
    END.
(wtchar) PROCEDURE (x,y,l,char);                           6C1X5
    %output a single character at the point x,y%
    LOCAL i;
    REF char;
    gpsend(tekgm);
    g4cdst(x,y);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekpad);
    gpsend(tekam);
    FOR i_1 UP UNTIL >l DO gpsend(*char*[i]);
    RETURN;
    END.
(wtarc) PROCEDURE (xcent,ycent,rad);                       6C1X6
```

```
        LOCAL radius,angle,x,y,xcnt,ycnt,i;
        gpsend(tekgm);
        xcnt _ float(xcent);
        ycnt _ float(ycent);
        radius _ float(rad);
        FOR i_0 UP 5 UNTIL >360 DO
           BEGIN
           angle _ fdrad(float(i));
           x _ fix(fadd(xcnt,fmul(radius,fcos(angle))));
           y _ fix(fadd(ycnt,fmul(radius,fsin(angle))));
           g4cdst(x,y);
           END;
        RETURN;
        END.
%FINISH.%%% % end of file 3 of support routines for split
```

```
version of subsystem%
% Interface with FE %                                          6C1Z
   (bldtrk)   %  build cursor tracking on or off command %
   PROCEDURE (switch, commands REF);                           6C1Z1
      % Procedure description
         FUNCTION
            Build command to close graphics terminal port and
            turn cursor tracking on or open the graphics
            terminal port and turn cursor tracking off.
            Append command to commands list.
         ARGUMENTS
            switch:  tracking on/off switch
               values: TRUE - tracking on, FALSE - tracking
               off
            commands:  address of commands list
         RESULTS
            none
         NON-STANDARD CONTROL
            none
         GLOBALS
            none
         %
      % Declarations %
         LOCAL procpar REF;
      %Build command%
         &procpar _ getlst(1);  %allocate list%
         #procpar# _ USE makedesc(uboole, switch, FALSE);
      %Add to commands list%
         addtobatch(&commands, &procpar, ctrack);
      % Return %
         RETURN;
      END.

   (gfbctl)   % manipulate global string for show commands %
   PROCEDURE (type, str REF);                                  6C1Z2
      % Procedure description
         FUNCTION
            Manipulate global string for show commands, i.e.
            add to it, initialize it or send it to FE for
            display in command feedback window via BE
            procedure FBCTL
         ARGUMENTS
            type:  type of function
               values:
                  typenullit - initialize global string
                  fbaddlit - add to global string
                  addcalit - send to FE and wait for user
                  response
            str:  address of string
         RESULTS
            none
         NON-STANDARD CONTROL
            none
         GLOBALS
            string whose address is in GSHWST
         %
```

```
      % Declarations %
         REF gshwst;
      % Perform requested function %
         CASE type OF
            = typenullit:  %initialize string%
               *gshwst* _ EOL;
            = fbaddlit:  %add to string%
               *gshwst* _ *gshwst*, *str*;
            = addcalit:  %send to FE and wait for user
            response%
               BEGIN
               fbctl(addcalit, &gshwst);
               gshwst.L _ 0;  %null out string%
               END;
            ENDCASE err($"Invalid type - gfbctl");
      % Return %
         RETURN;
      END.

   (dfeclw) PROCEDURE;  %clear graphics window in FE%
      REF garglist;
      #garglist# _ USE makedesc(uindex, gprwndw.widindex,
      FALSE);
      extcall(feident, $"clear-window", fedpypkg, &garglist);
      RETURN;
      END.
   (dfedew) PROCEDURE;  %delete graphics window in FE%
      REF garglist;
      #garglist# _ USE makedesc(uindex, gprwndw.widindex,
      FALSE);
      extcall(feident, $"delete-window", fedpypkg, &garglist);
      RETURN;
      END.
   (fbctl)   % GB: ;  Display string in the command feedback
   window %
   PROCEDURE (type, str REF);                                  6C1Z5
      % Procedure description
         FUNCTION
            Display string in the command feedback window and
            wait for user response if requested  This routine
            replaces the 8.5 fbctl (feed back control
            routine).
         ARGUMENTS
            type:  function type
               values:
               typenullit -- clear display in preparation for
               display
                  currently does not call the FE, i.e. results
                  in no action
               fbaddlit -- add string to display
               addcalit -- add "OK:" to display and wait for
               user response
               typelit -- clear display, display string and
               wait for user input
                  not implemented
               fbendlit -- add string to display and wait for
```

```
                    user input
                       not implemented
                    typecalit -- clear display, display string
                    adding phrase "OK:" and wait for user response
                       currently does not clear display
                 str:  address of string
              RESULTS
                 none
              NON-STANDARD CONTROL
                 Go to ERR if FE returns value FALSE indicating the
                 user did a command delete
              GLOBALS
                 none
              %
           % Declarations %
              LOCAL
                 confirm = TRUE,
                 noconfirm = FALSE;
           % call procedure to send string to FE %
              CASE type OF
                 = typenullit:  NULL;
                 = fbaddlit:
                    shcfw(&str, noconfirm);  %use string as is%
                 =addcalit, =typecalit:
                    shcfw(&str, confirm);  %use string as is%
                 ENDCASE
                    err($"type not implemented - fbctl");
           % Return %
              RETURN;
           END.

        (shcfw)   % LB: ;  display string in command feedback
        window %
        PROCEDURE (str REF, confirm);                              6C1Z6
           % Procedure description
              FUNCTION
                 Call FE procedure SHOW which displays string in
                 the command feedback window.  If string is too
                 long to be sent to FE in one call, this routine
                 breaks string into segments and does multiple
                 calls on FE by calling itself recursively.  This
                 routine assumes that string is allocated from the
                 list zone - it does not copy it in.
              ARGUMENTS
                 str:  address of string
                 confirm:  TRUE if user response requested, FALSE
                 if no response
              RESULTS
                 TRUE if no user response or user type <CA>
                 GLOBALS
              NON-STANDARD CONTROL
                 Abort if call on Fe fails
              GLOBALS
                 Uses maxdpschar
              %
           % Declarations %
```

```
            LOCAL proceed, endstr;
            LOCAL retlst REF _ 0;
            LOCAL TEXT POINTER tp1, tp2;
            LOCAL workstr REF _ 0;  %must be at least maxdpschar
            long%
            REF garglist;
         % Check if string can be sent in one call or not %
            IF str.L <= maxdpschar THEN
               BEGIN
               #garglist# _ USE makedesc (ustring, &str, FALSE),
                  USE makedesc (uboole, confirm, FALSE);
               IF NOT extcall(feident, $"SHOW", fetoolpkg,
               &garglist: &retlst) THEN
                  ABORT (erdfedisplay,$"Error in FE - shcfw");
               END
               ELSE  %break string up and send to FE%
                  BEGIN
                  &workstr _ galstr(maxdpschar);  %alloc. string%
                  FIND SF(*str*) ^tp1 ;
                  tp2 _ tp1;
                  tp2[1] _ tp1[1] + maxdpschar;
                  endstr _ str.L + 1;  %value for tp for last
                  segment%
                  WHILE tp2[1] < endstr DO   %Loop until entire
                  string sent%
                     BEGIN  %set work string to next maxdpschar
                     chars%
                     tp1[1] _ tp2[1] + 1;  %point to char. after
                     last sent%
                     tp2 [1] _ MIN (endstr, tp1[1] + maxdpschar);
                     *workstr* _ tp1 tp2;
                     %send segment to FE.  Use FALSE confirm flag
                     until last segment is sent.  Use real
                     confirm flag with last segment, i.e. don't
                     wait for response until entire string has
                     been and wait at that moment only if user
                     response is requested%
                        shcfw(&workstr, IF tp2[1] = endstr THEN
                        confirm ELSE FALSE);
                     END;
                  gfrstr(&workstr);  %free string%
                  END;
         % Get result from Fe  %
            IF retlst.L THEN proceed _ ELEM #retlst#[1]
            ELSE proceed _ TRUE;
         % Return %
            RETURN (proceed);
         END.

         FINISH
FINISH
```