UIS Source Code Annotator User's Guide

Order Number: AA-PBZTA-TE

Software Version: UIS Source Code Annotator V2.0

Operating System: VMS V5.1 or above

August 1990

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

Preface

The *UIS Source Code Annotator User's Guide* describes what the UIS Source Code Annotator is and how to use it.

You *must* install the UIS Source Code Annotator on VMS V5.1 or above.

Structure of this Manual

This manual describes what the UIS Source code Annotator is and how to use it.

This manual includes two chapters and seven appendixes.

Chapter 1	Describes what the Annotator does and it's characteristics.
Chapter 2	Describes how to use the Annotator.
Appendix A	Presents a UISANN$ROUTINES.TABLE.
Appendix B	List messages you may encounter while using the Annotator.
Appendix C	Lists UIS$ routines with their equivalent Xlib routines and describes their functionality.
Appendix D	Lists UISDC$ routines with their equivalent Xlib routines and describes their functionality.
Appendix E	Lists HCUIS$ routines with their equivalent Xlib routines and describes their functionality.
Appendix F	Presents two sample FORTRAN programs and a summary report.
Appendix G	Presents two sample PASCAL programs and a summary report.

Related Documents

If you want to migrate your VWS applications to DECwindows, consult the appropriate documents:

- *UIS Source Code Annotator User's Guide* for information about using the source code annotator.
- *Using the UIS to DDIF Converter* for information about using the UIS to DDIF Converter.
- *A Guide to Migrating VWS Applications to DECwindows* for information about migrating VWS applications to DECwindows and for an example application.
- *Using the DECwindows/X11 Server for VWS* for information about using the DECwindows/X11 Server for VWS.
- *VAXuisx User's Guide* for information about using the VAXuisx run-time library.

If you are working with VWS, you may wish to consult the following documents:

- *VMS Workstation Software User's Guide* for information about how to use the workstation software.
- *VMS Workstation Software Graphics Programming Guide* for information about working with application programs and using VMS Workstation Software graphics.
- *VMS Workstation Software Guide to Printing Graphics* for detailed information about how to print hard copies from the VAXstation.
- *VMS Workstation Software SIGHT User's Guide* for detailed information about using SIGHT.

Conventions

This manual uses the following conventions in user input examples.

Conventions and Meanings

RETURN

The RETURN key is not always shown in formats and examples. Assume that you must press the RETURN key after typing a command or other input to the system, unless you are instructed otherwise.

CTRL/X

CTRL followed by a slash and a letter means that you must type the letter while holding down the CTRL key. For example, CTRL/B means hold down the CTRL key and type the letter B.

Lists

When a format item is followed by a comma and an ellipsis (, . . .), you can enter a single item or a number of items separated by commas. When a format item is followed by a plus sign and an ellipsis (+ . . .), you can enter a single item or a number of those items connected by plus signs. If you enter a list (more than one item), you must enclose the list in parentheses. A single item need not be enclosed in parentheses.

Optional Items

An item enclosed in square brackets ([ ]) is optional.

Boxes

In examples, boxes enclose user input, such as a key 2, a key sequence CTRL/Z, or a parameter PASSWORD.

Ellipsis

.
.
.

A vertical ellipsis indicates that some of the format or example is not shown.

<⌫

The key on the LK201 terminal keyboard that performs the DELETE function is labeled <⌫.

1 Introduction to the UIS Source Code Annotator

This chapter describes the UIS Source Code Annotator (hereafter called the Annotator), what it does, the languages it uses and any restrictions that you may encounter when using the Annotator with VWS applications.

The Annotator provides information you can use in when transferring your UIS applications to DECwindows. By passing a UIS application through the Annotator, you can learn which UIS$ routines have equivalent Xlib routines and what those routines are. You also can learn which UIS$ routines have no equivalent Xlib routines. You should use the Annotator along with *A Guide to Migrating VWS Applications to DECwindows*.

NOTE: **VMS Workstation Software (VWS) is only supported on VMS. Consequently, the Annotator is only supported on VMS.**

VWS and the Annotator are not supported on ULTRIX.

1.1 Annotator Files

The Annotator consists of the following four files contained in the SYS$SYSROOT:[SYSHLP.EXAMPLES.UISANN] directory:

- **UISANN.EXE** - Executable image being run.
- **UISANN$ROUTINES.TABLE** - Routines table used for determining equivalences.
- **UISANN$IVP.COM** - Installation verification procedure (IVP).
- **UISANNMSG.EXE** - Message file.

The directory also contains sample programs that the IVP uses to test the Annotator after the installation. You can use these sample programs if you want to give the Annotator a "dry run." To see what the original files, annotated files, and summary reports, QIX.FOR and UISDC_HOUSE.PAS, should look like, refer to Appendix F and Appendix G.

1.2 Annotator Function

The Annotator assists you in converting a VWS application to DECwindows. For any VWS application you specify, except those written in Ada, the Annotator opens the file and searches the code. It flags every UIS$, UISDC$, and HCUIS$ routine, adding a comment to each routine that helps you port the application to DECwindows. The Annotator then produces the annotated and summary output files.

The annotated file contains the original code plus comments identifying which UIS$, UISDC$, and HCUIS$ routines have equivalent Xlib routines and which ones do not. When an equivalent or similar, Xlib routine exists, the Annotator lists that routine or an alternate method for acquiring the best result.

Note: The Annotator creates a new version of your file with a higher version number. If you are concerned about comments being added to your file, you may want to create a second copy of your file before running the Annotator. However, files created by the Annotator will be compilable.

In some cases, there is no equivalent routine; however, there are methods of producing the desired result. For more information refer to *A Guide to Migrating VWS Applications to DECwindows*.

The annotated file is still a usable UIS application. The comments the Annotator places in your file do not prevent the annotated file from compiling successfully. After you compile the annotated file, you can use the resulting .OBJ file to create a usable image.

The comments the Annotator puts into the file can be up to 125 characters long. If you print the annotated file on a printer or in a print mode less than 132 characters wide then the output may be truncated or wrapped. To make sure you do not lose any of the comments, you should print the file on a line printer or a laser printer using a landscape format. The command for printing in a landscape format on an LPS40 or LN03R is:

```
$ PRINT/PARAMETER=(DATA_TYPE=ANSI,PAGE_ORIENTATION=LANDSCAPE) -
      /QUEUE=queue_name filename.ext
```

The command for printing on a line printer is:

```
$ PRINT/QUEUE=queue_name filename.ext
```

The following is a sample of VWS code to be Annotated.

```
C Create the display and window.  Enable the window resize option.
C
        CALL UIS$GET_HW_COLOR_INFO('SYS$WORKSTATION',,
        1                          VCM_SIZE)
        VCM_SIZE=16
        IF (VCM_SIZE .EQ. 2) GOTO 55
        VCM_SIZE = VCM_SIZE/4
        IF (VCM_SIZE .LT. NUM_LINES) GOTO 55
        NUM_LINES = 20
55      VCM_ID = UIS$CREATE_COLOR_MAP(VCM_SIZE)
        VD_ID = UIS$CREATE_DISPLAY(WC_X1,WC_Y1,WC_X2,WC_Y2,
        1                          VP_WIDTH,VP_HEIGHT,VCM_ID)
        CALL UIS$DISABLE_DISPLAY_LIST(VD_ID)
        CALL CREATE_COLORS(VCM_SIZE,VD_ID)
        WD_ID = UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','QIX')
        CALL UIS$SET_RESIZE_AST(vd_id,wd_id,ENABLE_WINDOW_RESIZE,dummy,
        1       new_abs_x,new_abs_y,vp_width,vp_height)
```

The following example shows the same code after it has been annotated.

```
C Create the display and window.  Enable the window resize option.
C
C     %UIS%  Information is available through a number of individual
C     calls - Please see "Display Routines".
        CALL UIS$GET_HW_COLOR_INFO('SYS$WORKSTATION',,
        1                          VCM_SIZE)
        VCM_SIZE=16
        IF (VCM_SIZE .EQ. 2) GOTO 55
        VCM_SIZE = VCM_SIZE/4
        IF (VCM_SIZE .LT. NUM_LINES) GOTO 55
        NUM_LINES = 20
C     %UIS%  Color maps may be created by using the X$ALLOC_COLOR_CELLS.
55      VCM_ID = UIS$CREATE_COLOR_MAP(VCM_SIZE)
C     %UIS%  No equivalent routine exists.
        VD_ID = UIS$CREATE_DISPLAY(WC_X1,WC_Y1,WC_X2,WC_Y2,
        1                          VP_WIDTH,VP_HEIGHT,VCM_ID)
C     %UIS%  X11 provides no equivalents to the UIS$ display list routines.
        CALL UIS$DISABLE_DISPLAY_LIST(VD_ID)
        CALL CREATE_COLORS(VCM_SIZE,VD_ID)
C     %UIS%  Please see information on virtual displays.
        WD_ID = UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','QIX')
C     %UIS%  There are equivalent X events.
        CALL UIS$SET_RESIZE_AST(vd_id,wd_id,ENABLE_WINDOW_RESIZE,dummy,
        1       new_abs_x,new_abs_y,vp_width,vp_height)
```

After the Annotator produces the annotated file, you can consult *A Guide to Migrating VWS Applications* for help in converting the VWS application to a DECwindows application.

The Summary Report has the same name as the source code file, but has a .LOG extension. It provides the following information:

- The number of UIS$, UISDC$, and HCUIS$ routines found in the application.
- The number of routines that were valid.
- The number of routines that were invalid, (routines beginning with UIS$, UISDC$, or HCUIS$, but not part of VWS).
- The number of times each routine was called.

This is an example of a summary report.

```
    CIRCLE.LOG
    Date :  11-MAY-90,  Time : 13:20:59

    This report is the result of searching of the following files:
        CIRCLE.PAS

    searching for UIS calls within programs. A summary will
    appear at the end of this report.


    >>> Examining : DISK2:[SMITH]CIRCLE.PAS
=== Creating  : DISK2:[SMITH]CIRCLE.PAS

        Found:          1 - HCUIS$WRITE_DISPLAY
        No equivalent routine exists.

        Found:          1 - UIS$CIRCLE
        Filled circles are drawn using the X$DRAW_ARC routine.

        Found:          1 - UIS$CREATE_DISPLAY
        No equivalent routine exists.
```

```
Found:           1 - UIS$DELETE_DISPLAY
UIS$DELETE_DISPLAY is similar to X$CLOSE_DISPLAY.

* Total Lines read in :          37

* Total UIS calls (of any type) detected :         4

*** Summary Information ----------------------------------------------

   * Total UIS calls (of any type) - all files :           4
```

After running a file through the Annotator, you end up with three files in your directory: the source file, the annotated file, and the summary file. For example, if you ran the file QIX.FOR;3 through the Annotator, you would have the following files in your directory:

- QIX.FOR;3 - Source file
- QIX.FOR;4 - Annotated file
- QIX.LOG - Summary file

1.3 Languages the Annotator Accepts

The Annotator accepts VWS applications written in the following languages:

- BASIC
- BLISS
- COBOL
- LISP
- PL/I
- VAX C
- VAX FORTRAN
- VAX/MACRO
- VAX Pascal

Note: The Annotator does not accept applications written in Ada.

1.4 Restrictions on Using the Annotator with VWS Applications

To use the Annotator with the VWS sample applications provided with the VWS kit, you must have VMS V5.1 or above installed.

If you want to compile the two sample programs, QIX.FOR and UISDC_HOUSE.PAS, you will need the VAX FORTRAN and VAX Pascal compilers.

2 Using the Annotator

This chapter describes how to invoke and use the Annotator.

2.1 Invoking the Annotator

All the files necessary for running the Annotator can be found in SYS$SYSROOT:[SYSHLP.EXAMPLES.UISANN]. In addition to the executables, there is a table file with a list of all the UIS$, UISDC$, and HCUIS$ routine names. In order to run the annotator the logical name UISANN$TABLE must point at this file.

You may want to define a symbol in order to simplify running the Annotator. It would be best to have your System Manager define these on a system wide basis. If this is not possible, you could define them in your LOGIN.COM.

```
$ UISANN == "RUN SYS$COMMON:[SYSHLP.EXAMPLES.UISANN]UISANN"

$ DEFINE UISANN$TABLE -
        SYS$COMMON:[SYSHLP.EXAMPLES.UISANN]UISANN$ROUTINES.TABLE
```

If the logical is not defined, the Annotator will not work and an error message will be returned.

2.2 Using the Annotator

The Annotator file has the same name and extensions as the source file, but the version number is one higher. For example, if the file containing the source code was named QUANTUM.PAS;3, the file containing the annotated code would be named QUANTUM.PAS;4.

The Annotator produces a Summary Report file that has the same name as the source code file, but has a .LOG extension. The Summary Report for QUANTUM.PAS would be QUANTUM.LOG.

The source files you want to annotate must be in your current directory. The Annotator places the annotated files and the Summary Report in that directory. You also must have sufficient privileges to access the source files you want annotated and to access the directory they are in.

Follow this procedure to use the Annotator:

1 Invoke the Annotator .

2 Select from the Annotator menu the language of the file you want annotated.

3 Enter the filename and extension (eg. QIX.FOR).

4 Exit from the Annotator.

The remainder of this section takes you through a "dry run" of the Annotator. You will annotate the example file UISDC_HOUSE.PAS. To do the dry run, you must have UISDC_HOUSE.PAS in your current directory. If you do not, the Annotator will not be able to find the file.

1 Invoke the Annotator by typing one of the following:

a. If you or your system manager has defined a global symbol so UISANN can be issued, type:

```
$ UISANN
```

b. If no global symbol exists, type:

```
$ RUN SYS$COMMON:[SYSHLP.EXAMPLES.UISANN]UISANN
```

The Annotator initializes the table and then displays the Annotator menu, which lists the languages the Annotator supports.

The is an example of the Annotator menu.

```
Select file type to examine or enter ? for help:
   BASIC      = 1
   BLISS      = 2
   C          = 3
   COBOL      = 4
   FORTRAN    = 5
   LISP       = 6
   PASCAL     = 7
   PLI        = 8
   VAX/MACRO  = 9
  or enter 0 or <CTRL/Z> to exit.

File Type to be Examined:
```

2 To select the language of the file you want, type the number that corresponds to the language and press RETURN.

```
File Type to be Examined: 7
```

The Annotator displays a message confirming the language you have selected.

```
%UISANN-I-ALLSOUPAS, all source code is assumed to be Pascal
```

The Annotator then prompts you to enter the name of the file you want annotated.

3 Type UISDC_HOUSE.PAS and press RETURN.

```
Please enter a file name, or file name with Wild Cards (*). For Help,
Enter ?.

File Name(s) to be Examined : uisdc_house.pas
```

The Annotator displays a list of UIS$ routines it has found in the program. For each routine, it identifies one or more Xlib routines that are equivalent or informs you that it could not find one.

Note: For more information on the comments the Annotator adds to the file, refer to Appendix C of *A Guide to Migrating VWS Applications to DECwindows.*

This is a screen display for UISDC_HOUSE.PAS.

```
 >>> Examining : DISK2:[SMITH]UISDC_HOUSE.PAS
%UISANN-I-CREATING, Creating the next version
of'DISK2:[SMITH]UISDC_HOUSE.PAS'

 Found:          1 - UIS$CREATE_COLOR_MAP
Color maps may be created by using the X$ALLOC_COLOR_CELLS.

 Found:          1 - UIS$CREATE_DISPLAY
No equivalent routine exists.

 Found:          1 - UIS$CREATE_WINDOW
Please see information on virtual displays.

 Found:          1 - UIS$GET_DISPLAY_SIZE
This may be emulated using X$DISPLAY_WIDTH, X$DISPLAY_WIDTHMM,
X$DISPLAY_HEIGHT, and X$DISPLAY_HEIGHTMM.

 Found:          1 - UIS$SET_COLOR
UIS$SET_COLOR is equivalent to X$STORE_COLOR.

 Found:          1 - UIS$SET_COLORS
UIS$SET_COLORS is equivalent to X$STORE_COLORS.

 Found:          4 - UIS$SET_FILL_PATTERN
UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC.

 Found:          4 - UIS$SET_FONT
UIS$SET_FONT is similar to X$SET_FONT. The font ID is obtained from
X$LOAD_FONT.

 Found:          4 - UIS$SET_WRITING_INDEX
UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_GC.

 Found:          1 - UISDC$CIRCLE
UISDC$CIRCLE is similar to X$DRAW_ARC.

 Found:          1 - UISDC$LINE
UISDC$LINE is similar to X$DRAW_SEGMENT or X$DRAW_POINT.

 Found:          1 - UISDC$LINE_ARRAY
UISDC$LINE_ARRAY is similar to X$DRAW_SEGMENTS or X$DRAW_POINTS.

 Found:          4 - UISDC$PLOT
UISDC$PLOT is similar to X$DRAW_LINE, X$DRAW_LINES, or X$DRAW_POINT.

 Found:          1 - UISDC$PLOT_ARRAY
UISDC$PLOT_ARRAY is similar to X$DRAW_LINE, X$DRAW_LINES, or
X$DRAW_POINT.

 Found:          1 - UISDC$SET_CHAR_SIZE
X11 does not provide text scaling.

 Found:          1 - UISDC$TEXT
UISDC$TEXT is similar to X$DRAW_TEXT.
```

```
   * Total Lines read in :          142
   * Total UIS calls (of any type) detected :          25

*** Summary Information ----------------------------------------------
   * Total UIS calls (of any type) - all files :          25
```

The information the Annotator displays on the screen is identical to the information contained in the Summary Report for the program in UISDC_HOUSE.LOG.

4 Exit from the Annotator, by typing CTRL/Z and pressing RETURN or by typing 0 and pressing RETURN.

A UISANN$ROUTINES.TABLE

This appendix contains the UISANN$ROUTINES.TABLE that the Annotator uses.

```
** UISANN$ROUTINES.TABLE - used by the UIS Source Code Annotator
**
*******************************************************************************
**                 *
**     COPYRIGHT © 1989, 1990 BY                                  *
**       DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                      *
**                                                                           *
** THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED     *
** ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE     *
** INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER     *
** COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
** OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY     *
** TRANSFERRED.                                                              *
**                                                                           *
** THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE     *
** AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
** CORPORATION.                                                              *
**                                                                           *
** DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS     *
** SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
* *          *
*******************************************************************************
**
** A list of up to 512 UIS, UISDC and HCUIS  calls & messages.  This
** will support any computer language that follows the facname$ convention.
** ADA is not supported.
**
** The format is:
**
** **     = Comment
** call:  = Call name (TESTED FOR AS UPPER CASE - UISANN$ROUTINES always
**       does upper case testing)
** mess:  = A displayable (to the CRT or a file) message for the user
**
** NOTE: Upper or lower case may be used as needed.
**
**  This table is broken up into three sections.  The first section will
**  contain all the UIS$ calls, the second will contain all UISDC$
**  calls, and the third all HCUIS$ calls that may be found in a
**  user application.
**
**  The sections will contain the routines in alphabetical order.
**  This order does not need to be maintained.  New routines should
**  be added alphabetically.
**
**
**
**  UIS
** 001
call: HCUIS$BEGIN_TRANSLATOR
mess: No equivalent routine exists.
** 002
call: HCUIS$END_TRANSLATOR
mess: No equivalent routine exists.
** 003
call: HCUIS$READ_BUFFER
mess: No equivalent routine exists.
```

```
** 004
call: HCUIS$READ_DISPLAY
mess: No equivalent routine exists.
** 005
call: HCUIS$TRANSLATE
mess: No equivalent routine exists.
** 006
call: HCUIS$WRITE_BUFFER
mess: No equivalent routine exists.
** 007
call: HCUIS$WRITE_DISPLAY
mess: No equivalent routine exists.
** 008
call: UIS$BEGIN_SEGMENT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 009
call: UIS$CIRCLE
mess: Filled circles are drawn using the X$DRAW_ARC routine.
** 010
call: UIS$CLOSE_WINDOW
mess: UIS$CLOSE_WINDOW is equivalent to a SYS$EXIT system service call.
** 011
call: UIS$COPY_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 012
call: UIS$CREATE_COLOR_MAP
mess: Color maps may be created by using the X$ALLOC_COLOR_CELLS.
** 013
call: UIS$CREATE_COLOR_MAP_SEG
mess: UIS$CREATE_COLOR_MAP_SEG may be emulated using X$CREATE_COLORMAP.
** 014
call: UIS$CREATE_DISPLAY
mess: No equivalent routine exists.
** 015
call: UIS$CREATE_KB
mess: No Direct Replacement - Similar to X$SELECT_INPUT.
** 016
call: UIS$CREATE_TERMINAL
mess: No DECwindows support provided for this UIS call.
** 017
call: UIS$CREATE_TB
mess: X11 does not provide digitizer support.
** 018
call: UIS$CREATE_TRANSFORMATION
mess: X11 provides only a device dependent integer coordinate space.
** 019
call: UIS$CREATE_WINDOW
mess: Please see information on virtual displays.
** 020
call: UIS$DELETE_COLOR_MAP
mess: UIS$DELETE_COLOR_MAP is similar to X$FREE_COLORMAP.
** 021
call: UIS$DELETE_COLOR_MAP_SEG
mess: UIS$DELETE_COLOR_MAP_SEG is similar to X$FREE_COLORMAP.
** 022
call: UIS$DELETE_DISPLAY
mess: UIS$DELETE_DISPLAY is similar to X$CLOSE_DISPLAY.
** 023
call: UIS$DELETE_KB
mess: UIS$DELETE_KB is similar to X$SELECT_INPUT.
** 024
call: UIS$DELETE_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 025
call: UIS$DELETE_PRIVATE
mess: X11 provides no equivalents to the UIS$ display list routines.
** 026
call: UIS$DELETE_TB
mess: X11 does not provide digitizer support.
** 027
```

```
call: UIS$DELETE_TRANSFORMATION
mess: X11 provides only a device dependent integer coordinate space.
** 028
call: UIS$DELETE_WINDOW
mess: Please see information on virtual displays.
** 029
call: UIS$DISABLE_DISPLAY_LIST
mess: X11 provides no equivalents to the UIS$ display list routines.
** 030
call: UIS$DISABLE_KB
mess: UIS$DISABLE_KB is similar to X$SELECT_INPUT.
** 031
call: UIS$DISABLE_KB
mess: X11 does to provide digitizer support.
** 032
call: UIS$DISABLE_VIEWPORT_KB
mess: There is no concept of a virtual KB in X11.
** 033
call: UIS$ELLIPSE
mess: Filled ellipses are drawn using the X$DRAW_ARC routine.
** 034
call: UIS$ENABLE_DISPLAY_LIST
mess: X11 provides no equivalents to the UIS$ display list routines.
** 035
call: UIS$ENABLE_KB
mess: UIS$ENABLE_KB is remotely similar to X$SET_INPUT_FOCUS.
** 036
call: UIS$ENABLE_TB
mess: X11 does not provide digitizer support.
** 037
call: UIS$ENABLE_VIEWPORT_KB
mess: There is not equivalent routine - please see X$SELECT_INPUT.
** 038
call: UIS$END_SEGMENT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 039
call: UIS$ERASE
mess: UIS$ERASE is similar to X$CLEAR_AREA; will repaint using the BG
pixmap if one is declared.
** 040
call: UIS$EXECUTE
mess: X11 provides no equivalents to the UIS$ display list routines.
** 041
call: UIS$EXECUTE_DISPLAY
mess: X11 provides no equivalents to the UIS$ display list routines.
** 042
call: UIS$EXPAND_ICON
mess: UIS$EXPAND_ICON is similar to X$SET_WM_HINTS.
** 043
call: UIS$EXTRACT_HEADER
mess: X11 provides no equivalents to the UIS$ display list routines.
** 044
call: UIS$EXTRACT_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 045
call: UIS$EXTRACT_PRIVATE
mess: X11 provides no equivalents to the UIS$ display list routines.
** 046
call: UIS$EXTRACT_REGION
mess: X11 provides no equivalents to the UIS$ display list routines.
** 047
call: UIS$EXTRACT_TRAILER
mess: X11 provides no equivalents to the UIS$ display list routines.
** 048
call: UIS$FIND_PRIMITIVE
mess: X11 provides no equivalents to the UIS$ display list routines.
** 049
call: UIS$FIND_SEGMENT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 050
```

```
call: UIS$GET_ABS_POINTER_POS
mess: UIS$GET_ABS_POINTER_POS is equivalent to X$QUERY_POINTER.
** 051
call: UIS$GET_ALLIGNED_POS
mess: X11 does not provide text formatting functions.
** 052
call: UIS$GET_ARC_TYPE
mess: X11 does not provide inquiry functions for graphics context.
** 053
call: UIS$GET_BACKGROUND_INDEX
mess: X11 does Not provide inquiry functions for graphics context.
** 054
call: UIS$GET_BUTTONS
mess: UIS$GET_BUTTONS is similar to X$QUERY_POINTER.
** 055
call: UIS$GET_CHAR_ROTATION
mess: X11 does not provide text rotation.
** 056
call: UIS$GET_CHAR_SIZE
mess: X11 does not provide text scaling.
** 057
call: UIS$GET_CHAR_SLANT
mess: X11 does not provide text shearing (slant).
** 058
call: UIS$GET_CHAR_SPACING
mess: X11 does not provide text formatting functions.
** 059
call: UIS$GET_CLIP
mess: X11 does not provide inquiry functions for graphics context.
** 060
call: UIS$GET_COLOR
mess: UIS$GET_COLOR is similar to X$QUERY_COLOR.
** 061
call: UIS$GET_COLORS
mess: UIS$GET_COLORS is similar to X$QUERY_COLORS.
** 062
call: UIS$GET_CURRENT_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 063
call: UIS$GET_DISPLAY_SIZE
mess: This may be emulated using X$DISPLAY_WIDTH, X$DISPLAY_WIDTHMM,
X$DISPLAY_HEIGHT, and X$DISPLAY_HEIGHTMM.
** 064
call: UIS$GET_FILL_PATTERN
mess: X11 does not provide inquiry functions for graphics context.
** 065
call: UIS$GET_FONT
mess: X11 does to provide inquiry functions for graphics context.
** 066
call: UIS$GET_FONT_ATTRIBUTES
mess: UIS$GET_FONT_ATTRIBUTES is similar to X$QUERY_FONT.
** 067
call: UIS$GET_FONT_SIZE
mess: UIS$GET_FONT_SIZE is similar to X$TEXT_WIDTH and X$TEXT_EXTENT.
** 068
call: UIS$GET_HW_COLOR_INFO
mess: Information is available through a number of individual calls -
Please see "Display Routines".
** 069
call: UIS$GET_INTENSITIES
mess: UIS$GET_INTENSITIES is similar to X$QUERY_COLORS.
** 070
call: UIS$GET_INTENSITY
mess: UIS$GET_INTENSITY is similar to X$QUERY_COLOR.
** 071
call: UIS$GET_KB_ATTRIBUTES
mess: UIS$GET_KB_ATTRIBUTES is similar to X$GET_KEYBOARD_CONTROL; except
for up button transitions..
** 072
call: UIS$GET_LINE_STYLE
```

```
mess: X11 does not provide inquiry functions for graphics context.
** 073
call: UIS$GET_LINE_WIDTH
mess: X11 does not provide inquiry functions for graphics context.
** 074
call: UIS$GET_NEXT_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 075
call: UIS$GET_OBJECT_ATTRIBUTES
mess: X11 provides no equivalents to the UIS$ display list routines.
** 076
call: UIS$GET_PARENT_SEGMENT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 077
call: UIS$GET_POINTER_POSITION
mess: UIS$GET_POINTER_POSITION is similar to X$QUERY_POINTER.
** 078
call: UIS$GET_POSITION
mess: X11 doe not provide text formatting functions.
** 079
call: UIS$GET_PREVIOUS_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 080
call: UIS$GET_ROOT_SEGMENT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 081
call: UIS$GET_TB_INFO
mess: X11 does not provide digitizer support.
** 082
call: UIS$GET_TB_POSITION
mess: X11 does not provide digitizer support.
** 083
call: UIS$GET_TEXT_FORMATTING
mess: X11 does not provide text formatting functions.
** 084
call: UIS$GET_TEXT_MARGINS
mess: X11 does not provide text formatting functions.
** 085
call: UIS$GET_TEXT_PATH
mess: X11 does not provide text formatting functions.
** 086
call: UIS$GET_TEXT_SLOPE
mess: X11 does not provide text formatting functions.
** 087
call: UIS$GET_VCM_ID
mess: X11 has no equivalent function.
** 088
call: UIS$GET_VIEWPORT_ICON
mess: Please see the Guide to Xlib Programming, "Using Properties to
Communicate with the Window Manager".
** 089
call: UIS$GET_VIEWPORT_POSITION
mess: UIS$GET_VIEWPORT_POSITION is similar to X$GET_WINDOW_ATTRIBUTES; note
the change in origin.
** 090
call: UIS$GET_VIEWPORT_SIZE
mess: UIS$GET_VIEWPORT_SIZE is similar to X$GET_WINDOW_ATTRIBUTES; UIS
returns centipoints - X returns pixels.
** 091
call: UIS$GET_VISIBILITY
mess: There is no direct way of obtaining this information in X11.  Track
visibility notify events.
** 092
call: UIS$GET_WINDOW_ATTRIBUTES
mess: UIS$GET_WINDOW_ATTRIBUTES is similar to X$GET_WINDOW_ATTRIBUTES.
** 093
call: UIS$GET_WINDOW_SIZE
mess: UIS$GET_WINDOW_SIZE is similar to X$GET_GEOMETRY: please note the
differences.
** 094
```

```
call: UIS$GET_WRITING_INDEX
mess: X11 does not provide inquiry functions for graphics Context.
** 095
call: UIS$GET_WRITING_MODE
mess: X11 does not provide inquiry functions for graphics Context.
** 096
call: UIS$GET_WS_COLOR
mess: UIS$GET_WS_COLOR is similar to X$LOOKUP_COLOR.
** 097
call: UIS$GET_WS_INTENSITY
mess: UIS$GET_WS_INTENSITY is similar to X$LOOKUP_COLOR.
** 098
call: UIS$HLS_TO_RGB
mess: X11 does not provide color conversion routines.
** 099
call: UIS$HSV_TO_RGB
mess: X11 does not provide color conversion routines.
** 100
call: UIS$IMAGE
mess: UIS$IMAGE is similar to X$PUT_IMAGE: Note that image formats differ.
** 101
call: UIS$INSERT_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 102
call: UIS$LINE
mess: UIS$LINE requires X$DRAW_SEGMENT and X$DRAW_POINT (for the individual
points).
** 103
call: UIS$LINE_ARRAY
mess: UIS$LINE_ARRAY requires X$DRAW_SEGMENT and X$DRAW_POINT (for the
individual points).
** 104
call: UIS$MEASURE_TEXT
mess: UIS$MEASURE_TEXT is similar to X$QUERY_TEXT_EXTENTS. ( NOTE: This
assumes control list/attribute blocks were not used. )
** 105
call: UIS$MOVE_AREA
mess: UIS$MOVE_AREA is similar to X$COPY_AREA followed by an X$ERASE or
draw of filled rectangle over area.
** 106
call: UIS$MOVE_VIEWPORT
mess: UIS$MOVE_VIEWPORT is similar to X$MOVE_WINDOW.
** 107
call: UIS$MOVE_WINDOW
mess: There is no equivalent functionality under X11.
** 108
call: UIS$NEW_TEXT_LINE
mess: X11 does not provide text formatting functions.
** 109
call: UIS$PLOT
mess: UIS$PLOT is similar to X$DRAW_LINE, X$DRAW_LINES or X$DRAW_POINT.
** 110
call: UIS$PLOT_ARRAY
mess: UIS$PLOT_ARRAY is similar to X$DRAW_LINE, X$DRAW_LINES or X$DRAW_POINT.
** 111
call: UIS$POP_VIEWPORT
mess: UIS$POP_VIEWPORT is equivalent to X$RAISE_WINDOW.
** 112
call: UIS$PRESENT
mess: No equivalent call exists, but may be easily adapted in line for
DECwindows.
** 113
call: UIS$PRIVATE
mess: X11 provides no equivalents to the UIS$ display list routines.
** 114
call: UIS$PUSH_VIEWPORT
mess: UIS$PUSH_VIEWPORT is similar to X$LOWER_WINDOW.
** 115
call: UIS$READ_CHAR
mess: No equivalent call exists, keyboard input is delivered using the X
```

```
event mechanism.
** 116
call: UIS$RESIZE_WINDOW
mess: UIS$RESIZE_WINDOW is similar to X$CHANGE_WINDOW_ATTRIBUTES.
** 117
call: UIS$RESTORE_CMS_COLORS
mess: UIS$RESTORE_CMS_COLORS is similar to X$INSTALL_COLORMAP.
** 118
call: UIS$RGB_TO_HLS
mess: X11 does not provide color conversion routines.
** 119
call: UIS$RGB_TO_HSV
mess: X11 does not provide color conversion routines.
** 120
call: UIS$SET_ADDOPT_AST
mess: No equivalent routine exists.
** 121
call: UIS$SET_ALLIGNED_POSITION
mess: X11 does not provide text formatting functions.
** 122
call: UIS$SET_ARC_TYPE
mess: UIS$SET_ARC_TYPE is similar to X$SET_ARC_MODE or X$CHANGE_GC.
** 123
call: UIS$SET_BACKGROUND_INDEX
mess: UIS$SET_BACKGROUND_INDEX is similar to X$SET_BACKGROUND or X$CHANGE_GC.
** 124
call: UIS$SET_BUTTON_AST
mess: This is encompassed by the X event processing routines.
** 125
call: UIS$SET_CHAR_ROTATION
mess: X11 does not provide character rotation.
** 126
call: UIS$SET_CHAR_SIZE
mess: X11 does not provide text scaling.
**127
call: UIS$SET_CHAR_SLANT
mess: X11 does to provide character shearing (slant).
** 128
call: UIS$SET_CHAR_SPACING
mess: X11 does not provide text formatting functions.
** 129
call: UIS$SET_CLIP
mess: X$SET_CLIP_RECTANGLES provides UIS clipping and more.
** 130
call: UIS$SET_CLOSE_AST
mess: The only equivalent concept is encompassed by the DECwindows Toolkit.
** 131
call: UIS$SET_COLOR
mess: UIS$SET_COLOR is equivalent to X$STORE_COLOR.
** 132
call: UIS$SET_COLORS
mess: UIS$SET_COLORS is equivalent to X$STORE_COLORS.
** 133
call: UIS$SET_EXPAND_ICON_AST
mess: No equivalent routine exists.
** 134
call: UIS$SET_FILL_PATTERN
mess: UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC.
** 135
call: UIS$SET_FONT
mess: UIS$SET_FONT is similar to X$SET_FONT.  The font ID is obtained from
X$LOAD_FONT.
** 136
call: UIS$SET_GAIN_KB_AST
mess: There are equivalent X events for obtaining input focus.
** 137
call: UIS$SET_INSERTION_POSITION
mess: X11 provides no equivalents to the UIS$ display list routines.
** 138
```

```
call: UIS$SET_INTENSITIES
mess: UIS$SET_INTENSITIES is equivalent to X$STORE_COLORS.
** 139
call: UIS$SET_INTENSITY
mess: UIS$SET_INTENSITY is equivalent to X$STORE_COLOR.
** 140
call: UIS$SET_KB_AST
mess: There are equivalent X events; use X$CHANGE_WINDOW_ATTRIBUTES.
** 141
call: UIS$SET_KB_ATTRIBUTES
mess: UIS$SET_KB_ATTRIBUTES is similar to X$CHANGE_KB_CONTROL.
** 142
call: UIS$SET_KB_COMPOSE2
mess: X$SET_MODIFIER_MAPPING and X$CHANGE_KEYBOARD_MAPPING can remap the
keyboard input.
** 143
call: UIS$SET_KB_COMPOSE3
mess: X$SET_MODIFIER_MAPPING and X$CHANGE_KEYBOARD_MAPPING can remap the
keyboard input.
** 144
call: UIS$SET_KB_KEYTABLE
mess: X$CHANGE_KEYBOARD_MAPPING and X$SET_MODIFIER_MAPPING can remap the
keyboard input.
** 145
call: UIS$SET_LINE_STYLE
mess: UIS$SET_LINE_STYLE is similar to X$SET_LINE_ATTRIBUTES or X$CHANGE_GC.
** 146
call: UIS$SET_LINE_WIDTH
mess: UIS$SET_LINE_WIDTH is similar to X$SET_LINE_ATTRIBUTES or X$CHANGE_GC.
** 147
call: UIS$SET_LOSE_KB_AST
mess: There are equivalent X events; use X$CHANGE_WINDOW_ATTRIBUTES.
** 148
call: UIS$SET_MOVE_INFO_AST
mess: There are equivalent X events.
** 149
call: UIS$SET_POINTER_AST
mess: There are equivalent X events.
** 150
call: UIS$SET_POINTER_PATTERN
mess: UIS$SET_POINTER_POSITION is similar to X$DEFINE_CURSOR.
** 151
call: UIS$SET_POINTER_POSITION
mess: UIS$SET_POINTER_POSITION is similar to X$WARP_POINTER.
** 152
call: UIS$SET_POSITION
mess: X11 does not provide text formatting functions.
** 153
call: UIS$SET_RESIZE_AST
mess: There are equivalent X events.
** 154
call: UIS$SET_SHRINK_TO_ICON_AST
mess: There are equivalent X events.
** 155
call: UIS$SET_TB_AST
mess: No equivalent routine exists.
** 156
call: UIS$SET_TEXT_FORMATTING
mess: X11 does not provide text formatting functions.
** 157
call: UIS$SET_TEXT_MARGINS
mess: X11 does not provide text formatting functions.
** 158
call: UIS$SET_TEXT_PATH
mess: X11 does not provide text formatting functions.
** 159
call: UIS$SET_TEXT_SLOPE
mess: X11 does not provide text formatting functions.
** 160
call: UIS$SET_WRITING_INDEX
```

```
mess: UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_GC.
** 161
call: UIS$SET_WRITING_MODE
mess: UIS$SET_WRITING_MODE is similar to X$SET_FUNCTION or X$CHANGE_GC.
** 162
call: UIS$SHRINK_TO_ICON
mess: UIS$SHRINK_TO_ICON is similar to X$SET_WM_HINTS.
** 163
call: UIS$SOUND_BELL
mess: UIS$SOUND_BELL is similar to X$BELL.
** 164
call: UIS$SOUND_CLICK
mess: There is no way to sound the keyclick in X11.
** 165
call: UIS$TEST_KB
mess: The application should keep track of this through the X event mechanism.
** 166
call: UIS$TEXT
mess: UIS$TEXT is similar to X$DRAW_TEXT.
** 167
call: UIS$TRANSFORM_OBJECT
mess: X11 provides no equivalents to the UIS$ display list routines.
** 168
call: UISDC$ALLOCATE_DOP
mess: X11 does not have comparable interface to the hardware.
** 169
call: UISDC$CIRCLE
mess: UISDC$CIRCLE is similar to X$DRAW_ARC.
** 170
call: UISDC$ELLIPSE
mess: UISDC$ELLIPSE is similar to X$DRAW_ARC.
** 171
call: UISDC$ERASE
mess: UISDC$ERASE is similar to X$CLEAR_AREA or X$CLEAR_WINDOW; X will
repaint the BG using a pixmap if specified.
** 172
call: UISDC$EXECUTE_DOP_ASYNCH
mess: X11 does not have a comparable interface to the hardware.
** 173
call: UISDC$EXECUTE_DOP_ASYNCH
mess: X11 does not have a comparable interface to the hardware.
** 174
call: UISDC$EXECUTE_DOP_SYNCH
mess: X11 does not have a comparable interface to the hardware.
** 175
call: UISDC$GET_ALIGNED_POSITION
mess: X11 no text formatting nor the concept of a current text writing position.
** 176
call: UISDC$GET_CHAR_SIZE
mess: X11 does not provide text scaling.
** 177
call: UISDC$GET_CLIP
mess: X11 does not provide query routines for Graphics Context.
** 178
call: UISDC$GET_POINTER_POSITION
mess: UISDC$GET_POINTER_POSITION is similar to X$QUERY_POINTER.
** 179
call: UISDC$GET_POSITION
mess: X11 does not provide text formatting.
** 180
call: UISDC$GET_TEXT_MARGINS
mess: X11 provides no text formatting nor the concept of text margins.
** 181
call: UISDC$GET_VISIBILITY
mess: There is no direct way of obtaining this information in X11.  Track
visibility notify events.
** 182
call: UISDC$IMAGE
mess: UISDC$IMAGE is similar to X$PUT_IMAGE.
** 183
```

```
call: UISDC$LINE
mess: UISDC$LINE is similar to X$DRAW_SEGMENT or X$DRAW_POINT.
** 184
call: UISDC$LINE_ARRAY
mess: UISDC$LINE_ARRAY is similar to X$DRAW_SEGMENTS or X$DRAW_POINTS.
** 185
call: UISDC$LOAD_BITMAP
mess: No equivalent routine exists.
** 186
call: UISDC$MEASURE_TEXT
mess: UISDC$MEASURE_TEXT is similar to X$QUERY_TEXT_EXTENTS.  (Assumes a
UIS control list was not used.)
** 187
call: UISDC$MOVE_AREA
mess: UISDC$MOVE_AREA is similar to X$COPY_AREA followed by X$ERASE or draw
of filled rectangle of the area.
** 188
call: UISDC$NEW_TEXT_LINE
mess: X11 does not provide text formatting.
** 189
call: UISDC$PLOT
mess: UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or X$DRAW_POINT.
** 190
call: UISDC$PLOT_ARRAY
mess: UISDC$PLOT_ARRAY is similar to X$DRAW_LINE, X$DRAW_LINES, or X$DRAW_POINT.
** 191
call: UISDC$QUEUE_DOP
mess: No equivalent routine exists.
** 192
call: UISDC$READ_IMAGE
mess: UISDC$READ_IMAGE is similar to X$GET_IMAGE.  CAUTION:  The contents
of X11 windows are not guaranteed to be valid.
** 193
call: UISDC$SET_ALIGNED_POSITION
mess: X11 provides no text formatting nor the concept of a current text
writing position.
** 194
call: UISDC$SET_BUTTON_AST
mess: This function is encompassed by X event processing.
** 195
call: UISDC$SET_CHAR_SIZE
mess: X11 does not provide text scaling.
** 196
call: UISDC$SET_CLIP
mess: X$SET_CLIP_RECTANGLES provides UIS clipping and more.
** 197
call: UISDC$SET_POINTER_AST
mess: There are equivalent X events; use X$CHANGE_WINDOW_ATTRIBUTES.
** 198
call: UISDC$SET_POINTER_PATTERN
mess: UISDC$SET_POINTER_PATTERN is similar to X$DEFINE_CURSOR.
** 199
call: UISDC$SET_POINTER_POSITION
mess: UISDC$SET_POINTER_POSITION is similar to X$WARP_POINTER.
** 200
call: UISDC$SET_POSITION
mess: X11 does not provide text formatting.
** 201
call: UISDC$SET_TEXT_MARGINS
mess: X11 provides no text formatting nor the concept of text margins.
** 202
call: UISDC$TEXT
mess: UISDC$TEXT is similar to X$DRAW_TEXT.
```

B Annotator Messages

This appendix contains messages you may encounter while using the Annotator.

ALLSOUBAS, All source code is assumed to be BASIC

Informational: All source code within the specified file is expected to be BASIC.

User Action: None.

ALLSOUBLI, All source code is assumed to be BLISS

Informational: All source code within the specified file is expected to be BLISS.

User Action: None.

ALLSOUC, All source code is assumed to be C

Informational: All source code within the specified file is expected to be C.

User Action: None.

ALLSOUCOB, All source code is assumed to be COBOL

Informational: All the source code within the specified file is expected to be COBOL.

User Action: None.

ALLSOUFOR, All source code is assumed to be FORTRAN

Informational: All source code within the specified file is expected to be FORTRAN.

User Action: None.

ALLSOULIS, All source code is assumed to be LISP

Informational: All source code within the specified file is expected to be LISP.

User Action: None.

ALLSOUMAC, All source code is assumed to be VAX/MACRO

Informational: All source code within the specified file is expected to be MACRO.

User Action: None.

ALLSOUPAS, All source code is assumed to be PASCAL

Informational: All source code within the specified file is expected to be PASCAL.

User Action: None.

ALLSOUPLI, All source code is assumed to be PL/I

Informational: All source code within the specified file is expected to be PL/I.

User Action: None.

CANCRE, Cannot create the next version of '!AS'/FAO=1

Fatal Error: The attempt to create the next version of the input file has failed.

User Action: Make sure you have the proper privileges to create files in your current directory. If you do, make sure you you have not exceeded any of your quotas. Also make sure the current version number is not 32767.

CREATING, Creating the next version of'!AS'/FAO=1

Informational: The next version of the input file has been successfully created.

User Action: None.

FILERRVER, File error - Verify the file name and directory

Warning: The input file name or the directory specification provided is incorrect.

User Action: Verify the file name and the directory specification. If they are incorrect, reenter them correctly. If they are correct, check the file and directory protection to make sure you have access.

INITAB, Initializing the tables

Informational: The table of UIS routines is being initialized.

User Action: None.

INVFILNAM, Invalid file name or specification

Warning: The file name or the directory specification you provided did not exist or did not have the appropriate protection to allow access.

User Action: Verify the file name and directory specifications. If they are incorrect, reenter them correctly. If they are correct, check the file and directory protection to make sure you do have access.

INVINPNOT, Invalid input - Nothing entered

Warning: The input to a question was invalid.

User Action: Reread the question and the acceptable input and reenter your response.

INVINPTOO, Invalid input - Too much entered

Warning: The response supplied contained too much information.

User Action: Reread the question and the acceptable input and reenter your response.

INVLANOPT, Invalid language option selected

Warning: The language option selected is invalid.

User Action: Review the valid options and select the one you want or exit by entering CTRL/Z or 0.

NOMODFIL, No modified file will be produced

Informational: A user action has occurred that does not warrant annotation taking place.

User Action: Determine if you supplied incorrect information to one of the queries. If you have, rerun the annotator and provide the correct information. If you have not, no action is required.

NORMAL, Normal successful completion

Informational: The Annotator annotated the file successfully

User Action: None.

UNATOFIN, Unable to find UIS_X_FLAG.TABLE

Fatal Error: The Annotator did not find the table UISANN$ROUTINES.TABLE.

Use Action: Check the definition of the logical UISANN$TABLE to make sure it points to the directory that contains the file UISANN$ROUTINES.TABLE. If UISANN$TABLE is not defined, define it with the following command:

```
$ DEFINE UISANN$TABLE SYS$SYSROOT:[SYSHLP.EXAMPLES.UISANN].
```

Then ask your system manager to define it system wide.

UNRINPPRO, Unrecognized input provided

Warning: The input provided to a query is not recognized as a valid response.

User Action: Reread the question and the acceptable inputs and reenter your response.

C UIS$ Routines and Equivalent Xlib Routines

This appendix list UIS$ routines you may encounter with their equivalent XLIB routines and describes their functionality.

C.1 Introduction to UIS$ Routines

Table C–1 gives UIS$ routines with their equivalent Xlib routines, and an explanation of the routine functionality.

Note: If no equivalent Xlib routine exists for a UIS$ routine, the "Xlib Routines" column contains N/A.

Table C–1 UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$BEGIN_SEGMENT	N/A	X11 provides no equivalent to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$CIRCLE	X$DRAW_ARC	You use the Xlib draw arc routine to draw circles. Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming.*
UIS$CLOSE_WINDOW	SYS$EXIT	The CLOSE_WINDOW routine is equivalent to a SYS$EXIT system service call and is the default action for the UIS$CLOSE_AST.
UIS$COPY_OBJECT	N/A	X11 provides no equivalent to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$CREATE_COLOR_MAP	X$ALLOC_COLOR_CELLS	X11 incorporates the concept of *private* colormaps that can be created by the application and installed (refer to X$CREATE_COLORMAP). However, the installation of a colormap other than the default usually alters the described in "Using Color" in the *VMS DECwindows Guide to Xlib Programming*. In general, to allocate colors for exclusive use (that is, you intend to alter the color), use X$ALLOC_COLOR_CELLS, requesting N planes and 1 color. This provides you with a single pixel value and a set of plane mask bits that can then be permuted to form a colormap that maintains the ability to be complemented (when you use GXxor mode with the plane mask bits). If no arithmetic operations must be performed on the bitmap, make the call with 1 plane and N colors. This has a better chance of succeeding. For applications using static colors, you can request "named" colors such as "red".
UIS$CREATE_COLOR_MAP_SEG	X$CREATE_COLORMAP	To emulate this feature, you must create and install a private colormap for the entire hardware colormap, and the application must manage this colormap. Refer to "Using Color" in the *VMS DECwindows Guide to Xlib Programming*. Complete control over the entire colormap is the only way to accomplish this.
UIS$CREATE_DISPLAY	N/A	This routine has no counterpart in X11. VWS uses this routine to initialize structures and create any needed colormap. In an X11 application, this routine would be replaced by more generic application initialization.
UIS$CREATE_KB	X$SELECT_INPUT	X11 has no equivalent to the UIS virtual keyboard. In X11, you can select the types of input events. Refer to "Handling Events" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$CREATE_TERMINAL	N/A	The DECterm VT340 terminal emulation windows can only be created from the DECwindows session manager. No mechanism exists to create a terminal window from within a program.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$CREATE_TB	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.
UIS$CREATE_ TRANSFORMATION	N/A	Since X11 provides only a device-dependent integer coordinate space with each unit representing a pixel, programmers must provide their own world coordinates and transformations.
UIS$CREATE_WINDOW	X$CREATE_WINDOW	This routine performs a device assignment to the workstation screen. This is the equivalent of X$OPEN_DISPLAY, which establishes the link to the display. In addition, an X$CREATE_WINDOW would be performed to create and initialize the window structures; this would be followed by an X$MAP_WINDOW to make the window visible. Refer to "Managing the Client-Server Connection" and "Working with Windows" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$DELETE_COLOR_MAP	X$FREE_COLORMAP	Refer to "Using Color", in the *VMS DECwindows Guide to Xlib Programming*.
UIS$DELETE_COLOR_MAP_ SEGMENT	X$FREE_COLORMAP	Refer to "Using Color", in the *VMS DECwindows Guide to Xlib Programming*.
UIS$DELETE_DISPLAY	X$CLOSE_DISPLAY	Closing the display is the nearest equivalent under X11. The X$DESTROY_WINDOW call is similar to moving the viewport offscreen under VWS. It leaves everything set up (like the connection), but does not leave the window.
UIS$DELETE_KB	X$SELECT_INPUT	X11 provides no equivalent to a virtual keyboard. Types of input events can be selected and keyboard events ignored. Refer to "Handling Events", in the *VMS DECwindows Guide to Xlib Programming*.
UIS$DELETE_OBJECT	N/A	X11 provides no equivalent to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$DELETE_PRIVATE	N/A	X11 provides no equivalent to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$DELETE_TB	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.
UIS$DELETE_TRANSFORMATION	N/A	Since X11 provides only a device-dependent integer coordinate space with each unit representing a pixel, programmers must provide their own world coordinates and transformations.
UIS$DELETE_WINDOW	X$CLOSE_DISPLAY	Closing the display is the nearest equivalent under X11. The X$DESTROY_WINDOW call is more like moving the viewport offscreen under VWS. It leaves everything set up (like the connection), but does not leave the window.
UIS$DISABLE_DISPLAY_LIST	N/A	X11 provides no equivalent to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$DISABLE_KB	X$SELECT_INPUT	X11 provides no equivalent to a virtual keyboard. The types of input events can be selected and keyboard events ignored. Refer to "Handling Events", in the *VMS DECwindows Guide to Xlib Programming*.
UIS$DISABLE_TB	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.
UIS$DISABLE_VIEWPORT_TB	N/A	UIS uses this routine to disconnect a virtual KB from a window and remove the window from the list of windows that can be cycled to. X11 has no concept of a virtual KB. You either accept input focus and keyboard input events or do not choose to receive these events of the input focus. Refer to X$SELECT_INPUT.
UIS$ELLIPSE	X$DRAW_ARC	Draw ellipses by using the Xlib draw arc routine. Refer to "Drawing Graphics", in the *VMS DECwindows Guide to Xlib Programming*.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$ENABLE_DISPLAY_LIST	N/A	X11 provides no equivalent to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$ENABLE_KB	X$SET_INPUT_FOCUS	This call sets the input focus (the closest concept to connecting the physical keyboard to a window).
UIS$ENABLE_TB	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.
UIS$ENABLE_VIEWPORT_KB	N/A	UIS uses this routine to associate a virtual KB with a window and add the window to the list of windows that can be cycled to. X11 has no concept of a virtual KB. You either accept input focus and keyboard input events or do not choose to receive these events of the input focus. Refer to X$SELECT_INPUT.
UIS$END_SEGMENT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$ERASE	X$CLEAR_AREA	Both Clear Area and Clear Window routines are provided to erase portions of windows. You cannot use the Clear Area function on a PIXMAP. Instead, you should use a filled rectangle the size of the screen in the background color. Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$EXECUTE	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$EXECUTE_DISPLAY	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$EXPAND_ICON	X$SET_WM_HINTS	Generally, the user controls the state of the application window. To set the initial state of a window, use the property routines to communicate to the server. In addition, the server honors the hints after the window has been created and mapped. Thus, if you specify the Initial State for the window as X$C_NORMAL_STATE with the X$SET_WM_HINTS, a window currently in an icon state will be expanded.
UIS$EXTRACT_HEADER	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$EXTRACT_OBJECT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$EXTRACT_PRIVATE	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$EXTRACT_REGION	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$EXTRACT_TRAILER	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$FIND_PRIMITIVE	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$FIND_SEGMENT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$GET_ABS_POINTER_POS	X$QUERY_POINTER	This function returns the position of the pointer relative to the window as well as the current state of the modifier keys and buttons. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines".
UIS$GET_ALIGNED_POSITION	N/A	X11 does not provide text formatting functions.
UIS$GET_ARC_TYPE	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_BACKGROUND_INDEX	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_BUTTONS	X$QUERY_POINTER	This function returns the position of the pointer relative to the window as well as the current state of the modifier keys and buttons. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines".
UIS$GET_CHAR_ROTATION	N/A	X11 does not provide text rotation.
UIS$GET_CHAR_SIZE	N/A	X11 does not provide text scaling.
UIS$GET_CHAR_SPACING	N/A	X11 does not provide text formatting.
UIS$GET_CLIP	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_COLOR	X$QUERY_COLOR	Provides the RGB values for the specified index.
UIS$GET_COLORS	X$QUERY_COLORS	Provides the RGB values for the specified index values.
UIS$GET_CURRENT_OBJECT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$GET_DISPLAY_SIZE	Refer to "Explanation".	The X$DISPLAY_WIDTH, X$DISPLAY_WIDTH_MM, X$DISPLAY_HEIGHT, and X$DISPLAY_HEIGHT_MM calls provide the information needed to emulate this.
UIS$GET_FILL_PATTERN	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_FONT	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_FONT_ATTRIBUTES	X$QUERY_FONT	This routine, as well as X$LOOKUP_FONT_WITH_INFO, can return information associated with this call.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$GET_FONT_SIZE	X$QUERY_FONT	This routine, as well as X$LOOKUP_FONT_WITH_INFO, can return information associated with this call.
UIS$GET_HW_COLOR_INFO	N/A	The information returned by this call is available through a number of individual calls. Refer to the *Xlib Routines Reference Manual*, Part 1, "Display Routines".
UIS$GET_INTENSITIES	X$QUERY_COLORS	Use the X$QUERY_COLORS routines.
UIS$GET_INTENSITY	X$QUERY_COLOR	Use the X$QUERY_COLOR routines.
UIS$GET_KB_ATTRIBUTES	X$GET_KEYBOARD_ CONTROL	The attributes are not specified in the same manner but are available through this routine.
UIS$GET_LINE_STYLE	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_LINE_WIDTH	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_NEXT_OBJECT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$GET_OBJECT_ATTRIBUTES	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$GET_PARENT_SEGMENT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$GET_POINTER_POSITION	X$QUERY_POINTER	This function returns the position of the pointer relative to the window as well as the current state of the modifier keys and buttons. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines".
UIS$GET_POSITION	N/A	X11 does not provide text formatting functions.
UIS$GET_PREVIOUS_OBJECT	NA	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$GET_ROOT_SEGMENT	NA	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$GET_TB_INFO	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.
UIS$GET_TB_POSITION	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.
UIS$GET_TEXT_FORMATTING	N/A	X11 does not provide text formatting functions.
UIS$GET_TEXT_MARGINS	N/A	X11 does not provide text formatting functions.
UIS$GET_TEXT_PATH	N/A	X11 does not provide text drawing path (left-right) functions.
UIS$GET_TEXT_SLOPE	N/A	X11 does not provide text slope (rotation) functions.
UIS$GET_VCM_ID	N/A	X11 has no equivalent function. The colormap ID for X11 is the nearest equivalent and is returned when the colormap is created or the workstation default can be returned. In general, the X11 colormap is not equivalent to the UIS colormap.
UIS$GET_VIEWPORT_ICON	N/A	In general, icons are managed by the window manager. Communication and inquiry are performed via structures that provide "hints" to the window manager. Refer to "Using Properties" in the *Guide to Xlib Programming* to Communicate with the Window Manager. The X$SET_WM_HINTS routine contains an ICON WINDOW field that you can optionally use to supply a window that serves as the icon. This window ID is user-created. Normally, icons are supplied when you provide a PIXMAP to be used as the icon display data.
UIS$GET_VIEWPORT_POSITION	X$GET_WINDOW_ ATTRIBUTES	You can obtain a data structure that provides information about the current position size and other window attributes.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$GET_VIEWPORT_SIZE	X$GET_WINDOW_ ATTRIBUTES	You can obtain a data structure that provides information about the current position size and other window attributes.
UIS$GET_VISIBILITY	N/A	A direct method of obtaining this information does not exist in X11. Since the X11 application can be notified of all requests to expose a window and can be notified (after the fact) of any window occlusion, the application can therefore keep track of the current state of visibility.
UIS$GET_WINDOW_ATTRIBUTES	X$GET_WINDOW_ ATTRIBUTES	You can obtain a data structure that provides information about the current position size and other window attributes.
UIS$GET_WINDOW_SIZE	X$GET_GEOMETRY	You can obtain a data structure that provides information about the current position size and other window attributes.
UIS$GET_WRITING_INDEX	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_WRITING_MODE	N/A	X11 does not provide inquiry functions for GCs (the equivalent of UIS ATBs).
UIS$GET_WS_COLOR	X$LOOKUP_COLOR	DECwindows contains a set of named colors. This call returns the closest RGB values available for the hardware, as well as the ideal RGB values for the specified color. Appendix C of the *Guide to Xlib Programming* provides the names of the predefined colors for DECwindows.
UIS$GET_WS_INTENSITY	X$LOOKUP_COLOR	The intensity is returned as RGB values. You can use NTSC to convert the RGB values to an intensity. DECwindows contains a set of named colors. This call returns the closest RGB values available for the hardware, as well as the ideal RGB values for the specified color. Appendix C of the *Guide to Xlib Programming* provides the names of the predefined colors for DECwindows.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$HLS_TO_RGB	N/A	The X11 RGB system is based on a 16-bit integer value, while the UIS RGB system uses a floating point between 0 and 1. HLS conversion routines are widely available, and one is included in Appendix C of *A Guide to Migrating VWS Applications to DECwindows*.. Xlib libraries provide no conversion routines.
UIS$HSV_TO_RGB	N/A	The X11 RGB system is based on a 16-bit integer value, while the UIS RGB system uses a floating point between 0 and 1. HLS conversion routines are widely available, and one is included in Appendix C of *A Guide to Migrating VWS Applications to DECwindows*. Xlib libraries provide no conversion routines.
UIS$IMAGE	X$PUT_IMAGE	Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*. Note that you may have to reformat image data unless you create and install a colormap for images greater than 1 bit deep.
UIS$INSERT_OBJECT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$LINE	X$DRAW_SEGMENT	The X$DRAW_POINT routine is also used to draw individual points (zero length lines). Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$LINE_ARRAY	X$DRAW_SEGMENTS	The X$DRAW_POINTS routine is also used to draw individual points (zero length lines). Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$MEASURE_TEXT	X$QUERY_TEXT_EXTENTS	X11 provides an equivalent function to measure the length of a text string. Note that control strings and text formatting are not provided for text output.
UIS$MOVE_AREA	X$COPY_AREA	This is equivalent to an X$COPY_AREA followed by one or more X$CLEAR_AREA operations to clear the area no longer covered by the area moved. Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$MOVE_VIEWPORT	X$MOVE_WINDOW	This function changes the location of the window on the screen. In X11, this function can move the window partially offscreen. This feature is not possible with the UIS call.
UIS$MOVE_WINDOW	N/A	X11 provides no equivalent function, since this relocates the display list. When no display list is used, it works much like UIS$MOVE_AREA.
UIS$NEW_TEXT_LINE	N/A	X11 does not provide text formatting functions.
UIS$PLOT	X$DRAW_LINE	The X$DRAW_POINT routine is also used to draw individual points (zero length lines). Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$PLOT_ARRAY	X$DRAW_LINES	The X$DRAW_POINTS routine is also used to draw individual points (zero length lines). Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming*.
UIS$POP_VIEWPORT	X$RAISE_WINDOW	These are directly equivalent.
UIS$PRESENT	N/A	DECwindows applications are generally started with SYS$OUTPUT and given a device class of DC$_WORKSTATION (device controller type WS). Applications should first check for this device class as SYS$OUTPUT. If the class is not DC$_WORKSTATION, the application should check for a logical name DECW$DISPLAY to be defined. If this logical is present, the X$OPENDISPLAY call uses this as the display destination. If both of these options fail, and your application supports both UIS and DECwindows, you can call UIS$PRESENT to see if UIS is available.
UIS$PRIVATE	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.
UIS$PUSH_VIEWPORT	X$LOWER_WINDOW	These are directly equivalent.
UIS$READ_CHAR	N/A	Keyboard input is delivered via the X EVENT mechanism.
UIS$RESIZE_WINDOW	X$CHANGE_WINDOW_ ATTRIBUTES	You can use this call to resize the X11 window.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$RESTORE_CMS_COLORS	X$INSTALL_COLORMAP	This X11 function installs a colormap. When you use a private colormap, you can use this function to do the binding to the hardware. Note that all colors are affected by this call.
UIS$RGB_TO_HLS	N/A	The X11 RGB system is based on a 16-bit integer value, while the UIS RGB system uses a floating point between 0 and 1. HLS conversion routines are widely available, and one is included in Appendix C of *A Guide to Migrating VWS Applications to DECwindows*. Xlib libraries provide no conversion routines.
UIS$RGB_TO_HSV	N/A	The X11 RGB system is based on a 16-bit integer value, while the UIS RGB system uses a floating point between 0 and 1. HLS conversion routines are widely available, and one is included in Appendix C of *A Guide to Migrating VWS Applications to DECwindows*. Xlib libraries provide no conversion routines.
UIS$SET_ADDOPT_AST	N/A	DECwindows has no additional options selection. Use the DECwindows Toolkit and the appropriate widget set to find equivalent functionality.
UIS$SET_ALIGNED_POSITION	N/A	X11 does not provide text formatting functions.
UIS$SET_ARC_TYPE	X$SET_ARC_MODE	Most of the ARC drawing styles are available in X11.
UIS$SET_BACKGROUND_INDEX	X$SET_BACKGROUND	This is provided by the appropriate GC creation or modification command. Refer to the *Xlib Routines Reference Manual*, Part 1, "Graphics Context Routines". The background index is specified in the BACKGROUND value in the GC Values data structure.
UIS$SET_BUTTON_AST	N/A	This is included in X EVENT processing. Refer to the *Xlib Reference Manual*, Part 1, "Event Routines".
UIS$SET_CHAR_ROTATION	N/A	X11 does not provide character rotation.
UIS$SET_CHAR_SIZE	N/A	X11 does not provide character scaling.
UIS$SET_CHAR_SLANT	N/A	X11 does not provide character slanting.
UIS$SET_CHAR_SPACING	N/A	X11 does not provide text formatting functions.
UIS$SET_CLIP	X$SET_CLIP_RECTANGLES	This provides a superset of UIS clipping.
UIS$SET_CLOSE_AST	N/A	The DECwindows Toolkit contains the only equivalent concept in DECwindows.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$SET_COLOR	X$STORE_COLOR	X$STORE_COLOR sets the RGB value in a previously allocated color cell. The RGB values must be converted into 16-bit integer values. Refer to the *Xlib Reference Manual*, Part 1, "Color Routines".
UIS$SET_COLORS	X$STORE_COLORS	X$STORE_COLORS sets RGB values in a list of previously allocated color cells. The RGB values must be converted into 16-bit integer values. Refer to the *Xlib Reference Manual*, Part 1, "Color Routines".
UIS$SET_EXPAND_ICON_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines", for more information.
UIS$SET_FILL_PATTERN	X$SET_STIPPLE	UIS fill patterns are the equivalent of stipple patterns in X11. A stipple is a single-bit deep PIXMAP. The PIXMAP must be created and the pattern drawn into it. Usually this is accomplished with the X$PUT_IMAGE operation. The stipple can then be used in a GC as a pattern or mask.
UIS$SET_FONT	X$SET_FONT	This routine sets a font ID into a GC. You must use the X$LOAD_FONT routine to obtain the font ID. Refer to the *Xlib Routines Reference Manual*, Part 1, "Graphics Context Routines" and Part 2, "Font Routines" for information on these routines.
UIS$SET_GAIN_KB_AST	N/A	Equivalent X EVENTS exist for obtaining INPUT FOCUS. Refer to "Handling Events" in the *Guide to Xlib Programming.*
UIS$SET_INSERTION_POSITION	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$SET_INTENSITIES	X$STORE_COLOR	X$STORE_COLOR sets an RGB value in a previously allocated color cell. RGB values must be converted into 16-bit integer values. Derive RGB values by using the intensity value for each of the RGB components. Refer to the *Xlib Routines Reference Manual*, Part 1, "Color Routines".
UIS$SET_INTENSITY	X$STORE_COLORS	X$STORE_COLORS sets a list of RGB values in a list of previously allocated color cells. RGB values must be converted into 16-bit integer values. Derive RGB values by using the intensity value for each of the RGB components. Refer to the *Xlib Routines Reference Manual*, Part 1, "Color Routines".
UIS$SET_KB_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, Window Routines" for more information.
UIS$SET_KB_ATTRIBUTES	X$CHANGE_KEYBOARD_ CONTROL	The KB can be remapped as appropriate. Note that this is done in a completely different fashion in X11. Refer to the *Xlib Routines Reference Manual*, Part 2, Window and Session Manager Routines" for more information.
UIS$SET_KB_COMPOSE2	X$SET_MODIFIER_MAPPING	Along with the X$CHANGE_KEYBOARD_MAPPING routine, this can remap the keyboard input. Refer to the *Xlib Routines Reference Manual*, Part 2, Window and Session Manager Routines" for more information.
UIS$SET_KB_COMPOSE3	X$SET_MODIFIER_MAPPING	Along with the X$CHANGE_KEYBOARD_MAPPING routine, this can remap the keyboard input. Refer to the *Xlib Routines Reference Manual*, Part 2, "Window and Session Manager Routines" for more information.
UIS$SET_KB_KEYTABLE	X$CHANGE_KEYBOARD_ MAPPING	Along with the X$SET_KEYBOARD_MAPPING routine, this can remap the keyboard input. Refer to the *Xlib Routines Reference Manual*, Part 2, Window and Session Manager Routines" for more information.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$SET_LINE_STYLE	X$SET_LINE_ATTRIBUTES	Refer to the *Xlib Routines Reference Manual*, Part 1, Graphic Context Routines" for more information.
UIS$SET_LINE_WIDTH	X$SET_LINE_ATTRIBUTES	Refer to the *Xlib Routines Reference Manual*, Part 1, "Graphic Context Routines" for more information.
UIS$SET_LOSE_KB_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, Window Routines" for more information.
UIS$SET_MOVE_INFO_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines" for more information.
UIS$SET_POINTER_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines" for more information.
UIS$SET_POINTER_PATTERN	X$DEFINE_CURSOR	Refer to the *Xlib Routines Reference Manual*, Part 2, "Cursor Routines", for more information.
UIS$SET_POINTER_POSITION	X$WARP_POINTER	Refer to the *Xlib Routines Reference Manual*, Part 2, "Window and Session Manager Routines" for more information.
UIS$SET_POSITION	N/A	X11 does not provide text formatting functions.
UIS$SET_RESIZE_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines" for more information.
UIS$SET_SHRINK_TO_ICON_AST	N/A	You can obtain equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines" for more information.
UIS$SET_TB_AST	N/A	X11 provides no digitizer support. Tablets are supported only as replacements for the mouse.

Table C–1 (Cont.) UIS Routines and their Equivalent Xlib Routines

UIS$ Routines	Xlib Routines	Explanation
UIS$SET_TEXT_FORMATTING	N/A	X11 does not provide text formatting functions.
UIS$SET_TEXT_MARGINS	N/A	X11 does not provide text formatting functions.
UIS$SET_TEXT_PATH	N/A	X11 does not provide text formatting functions.
UIS$SET_TEXT_SLOPE	N/A	X11 does not provide text formatting functions.
UIS$SET_WRITING_MODE	X$SET_FUNCTION	This is provided by the appropriate GC creation or modification command. Refer to the *Xlib Routines Reference Manual*, Part 1, "Graphics Context Routines" for more information. The FUNCTION is the actual logical operator used for the operation. UIS "modes" are a combination of FUNCTION, FILL STYLE, FILL STIPPLE, FOREGROUND, and BACKGROUND pixel values. A routine that shows the mapping for most UIS writing modes is provided in Appendix G of *A Guide to Migrating VWS Applications to DECwindows*.
UIS$SHRINK_TO_ICON	X$SET_WM_HINTS	The state of the application is generally controlled exclusively by the user. Set the initial state of a window by using the property routines to communicate to the server. In addition, the server honors the hints after the window has been created and mapped. Thus, if you specify the Initial State for the window as X$C_ICONIC_STATE, a window currently in a window state will be iconified.
UIS$SOUND_BELL	X$BELL	Refer to the *Xlib Routines Reference Manual*, Part 2, "Window and Session Manager Routines" for more information.
UIS$SOUND_CLICK	N/A	The keyclick cannot be sounded in X11.
UIS$TEST_KB	N/A	Applications should keep track of this through the X EVENT mechanism for INPUT focus gain and lose events.
UIS$TEXT	X$DRAWTEXT	X11 routines doe not provide any of the text formatting or control lists provided by UIS.
UIS$TRANSFORM_OBJECT	N/A	X11 provides no equivalents to the UIS$ display list routines. Programmers must supply their own display list routines or reprogram in a higher-level graphic interface such as GKS or PHIGS.

D UISDC$ Routines and Equivalent Xlib Routines

This appendix lists UISDC$ routines with their equivalent Xlib routines and describes their functionality.

D.1 Introduction to UISDC$ Routines

In addition to the world coordinate interface (UIS), VWS provides a device-coordinate, or pixel-level, interface (UISDC) to the graphics system services.

When an application programs in device coordinates, it must make mixed use of UIS$ and UISDC$ routines. Only UIS$ routines that use or modify world coordinate positions are duplicated as UISDC$ routines. Most informational, attribute, windowing, and display routines exist only in UIS format and are shared by the two programming levels.

Table D–1 gives UISDC$ routines with their equivalent Xlib routines, and an explanation of the routine functionality.

NOTE: If an equivalent Xlib routine does not exist, this is indicated in the table by N/A.

Table D–1 UISDC$ Routines and their Equivalent Xlib Routines

UISDC$ Routines	Xlib Routines	Explanation
UISDC$ALLOCATE_DOP	N/A	The DOP interface is a device-dependent mechanism that queues drawing packets to the VSII/GPX and VS2000/GPX. No comparable hardware interface exists under X11.
UISDC$CIRCLE	X$DRAW_ARC	Use the Xlib draw arc routine to draw circles. Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming* for more information.
UISDC$ELLIPSE	X$DRAW_ARC	Use the Xlib draw arc routine to draw ellipses. Refer to "Drawing Graphics in the *VMS DECwindows Guide to Xlib Programming* for more information.

Table D–1 (Cont.) UISDC$ Routines and their Equivalent Xlib Routines

UISDC$ Routines	Xlib Routines	Explanation
UISDC$ERASE	X$CLEAR_AREA	Clear Area and Clear Window routines are both provided to erase portions of windows. Note that you cannot use the Clear Area function on a PIXMAP; instead, a filled rectangle the size of the screen in the background color is also equivalent. Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming* for more information.
UISDC$EXECUTE_DOP_ASYNCH	N/A	The DOP interface is a device-dependent mechanism that queues drawing packets to the VSII/GPX and VS2000/GPX. No comparable hardware interface exists under X11.
UISDC$EXECUTE_DOP_SYNCH	N/A	The DOP interface is a device-dependent mechanism that queues drawing packets to the VSII/GPX and VS2000/GPX. No comparable hardware interface exists under X11.
UISDC$GET_ALIGNED_ POSITION	N/A	X11 provides no text formatting or the concept of a current text-writing position.
UISDC$GET_CHAR_SIZE	N/A	X11 does not provide text scaling.
UISDC$GET_CLIP	N/A	X11 does not provide query routines for GCs.
UISDC$GET_POINTER_ POSITION	X$QUERY_POINTER	This function returns the position of the pointer relative to the window. It also returns the the current state of the modifier keys and buttons. Refer to the *Xlib Routines Reverence Manual*, Part 1, "Window Routines".
UISDC$GET_POSITION	N/A	X11 provides no text formatting.
UISDC$GET_TEXT_MARGINS	N/A	X11 provides no text formatting or the concept of text margins.
UISDC$GET_VISIBILITY	N/A	Since an X11 application can be notified of all requests to expose a window and can be notified of the occluding of a window after the fact, there is no direct way to obtain this information. However, the application can keep track of the current state of visibility.

Table D–1 (Cont.) UISDC$ Routines and their Equivalent Xlib Routines

UISDC$ Routines	Xlib Routines	Explanation
UISDC$IMAGE	X$PUT_IMAGE	Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming.* Note that image data may require reformatting unless you create and install a colormap for images greater than 1 bit deep.
UISDC$LINE	X$DRAW_SEGMENT	Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming* for more information.
UISDC$LINE_ARRAY	X$DRAW_SEGMENTS	Refer to "Drawing" Graphics in the *VMS DECwindows Guide to Xlib Programming* for more information.)
UISDC$LOAD_BITMAP	N/A	This routine loads a user bitmap into offscreen video memory. In some ways, this is similar to the X11 concept of a PIXMAP, but the concepts differ. The principal use for this under UIS is to load font data for drawing with DOPs.
UISDC$MEASURE_TEXT	X$QUERY_TEXT_EXTENTS	X11 provides an equivalent function to measure the length of a text string. Note that control strings and text formatting are not provided for text output.
UISDC$MOVE_AREA	X$COPY_AREA	Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming* for more information.
UISDC$NEW_TEXT_LINE	N/A	X11 provides no text formatting.
UISDC$PLOT	X$DRAW_LINE	Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming* for more information.
UISDC$PLOT_ARRAY	X$DRAW_LINES	Refer to "Drawing Graphics" in the *VMS DECwindows Guide to Xlib Programming* for more information.
UISDC$QUEUE_DOP	N/A	The DOP interface is a device-dependent mechanism that queues drawing packets to the VSII/GPX and VS2000/GPX. No comparable hardware interface exists under X11.
UISDC$READ_IMAGE	X$GET_IMAGE	Since the bitmap contents are not guaranteed under X11, be extremely cautious when you use this function.
UISDC$SET_ALIGNED_POSITION	N/A	X11 provides neither text formatting nor the concept of a current text-writing position.
UISDC$SET_BUTTON_AST	N/A	This is part of X EVENT processing. Refer to the *Xlib Routines Reference Manual*, Part 1, "Event Routines".

Table D–1 (Cont.) UISDC$ Routines and their Equivalent Xlib Routines

UISDC$ Routines	Xlib Routines	Explanation
UISDC$SET_CHAR_SIZE	N/A	X11 does not provide text scaling.
UISDC$SET_CLIP	X$SET_CLIP_RECTANGLES	X11 provides a superset of UIS clipping.
UISDC$SET_POINTER_AST	N/A	You can accomplish equivalent X EVENTS by using the EVENT MASK in the X$CHANGE_WINDOW_ATTRIBUTES call. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines", for more information.
UISDC$SET_POINTER_PATTERN	X$DEFINE_CURSOR	Refer to the *Xlib Routines Reference Manual*, Part 2, "Cursor Routines", for more information.
UISDC$SET_POINTER_POSITION	X$QUERY_POINTER	This function returns the position of the pointer relative to the window; it also returns the current state of the modifier keys and buttons. Refer to the *Xlib Routines Reference Manual*, Part 1, "Window Routines".
UISDC$SET_POSITION	N/A	X11 provides no text formatting.
UISDC$SET_TEXT_MARGINS	N/A	X11 provides no text formatting or the concept of text margins.
UISDC$TEXT	X$DRAW_TEXT	The X11 routines do not provide any of the text formatting or control lists provided by UIS.

E HCUIS$ Routines and Equivalent Xlib Routines

This appendix lists HCUIS$ routines with their equivalent Xlib routines and describes thier funtionality.

E.1 Introduction to HCUIS$ Routines

Table E–1 gives HCUIS$ routines with their equivalent Xlib routines, and an explanation of the routine functionality.

Note: If an equivalent Xlib routine does not exist, this is indicated in the table by N/A.

Table E–1 HCUIS$ Routines and their Equivalent Xlib Routines

HCUIS$ Routines	Xlib Routines	Explanation
HCUIS$BEGIN_TRANSLATOR	N/A	DECwindows does not support a hardcopy library.
HCUIS$END_TRANSLATOR	N/A	DECwindows does not support a hardcopy library.
HCUIS$RED_BUFFER	N/A	DECwindows does not support a hardcopy library.
HCUIS$READ_DISPLAY	N/A	DECwindows does not support a hardcopy library.
HCUIS$TRANSLATE	N/A	DECwindows does not support a hardcopy library.
HCUIS$WRITE_BUFFER	N/A	DECwindows does not support a hardcopy library.
HCUIS$WRITE_DISPLAY	N/A	DECwindows does not support a hardcopy library.

F Sample FORTRAN Program (QIX.FOR)

This appendix contains two versions of a sample FORTRAN program and a summary report. The first version is the original program before the user ran it through the Annotator. The second version is the annotated program. The Annotator also produced the summary report for the program it annotated.

F.1 The Original FORTRAN Program

This section shows the original FORTRAN program before the user ran it through the Annotator:

```
        PROGRAM QIX
c ************************************************************************
c *                                                                      *
c *  COPYRIGHT © 1983, 1985, 1986, 1987 BY                               *
c *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTTS.             *
c *  ALL RIGHTS RESERVED.                                                *
c *                                                                      *
c *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND      *
c *  COPIED ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE      *
c *  AND WITH  THE  INCLUSION  OF  THE ABOVE COPYRIGHT NOTICE. THIS      *
c *  SOFTWARE OR  ANY  OTHER COPIES  THEREOF MAY NOT BE PROVIDED OR      *
c *  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON.  NO TITLE TO AND      *
c *  OWNERSHIP OF  THE  SOFTWARE IS  HEREBY TRANSFERRED.                 *
c *                                                                      *
c *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT      *
c *  NOTICE  AND  SHOULD  NOT  BE  CONSTRUED  AS  A  COMMITMENT  BY      *
c *  DIGITAL EQUIPMENTCORPORATION.                                       *
c *                                                                      *
c *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY      *
c *  OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.      *
c *                                                                      *
c *                                                                      *
c ************************************************************************

!++
! FACILITY:
!    Qix demo for VaxStation II
!
! ABSTRACT:
!    This program is an animation demo that moves a bunch of lines around
!   within the window.  There are 10 lines.  The line currently at the front
!   of the bunch is repeatedly erased then redrawn at the rear of the bunch
!   to create the illusion that the lines are moving.  When the lines
!   hit any side of the window, they are deflected off.
!
!
!--

!    Implicit inputs
!
        IMPLICIT INTEGER(A-Z)

!    Include files
!
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
        INCLUDE 'SYS$LIBRARY:UISENTRY'
```

```
!    Declare AST routines as external
!
        EXTERNAL enable_window_resize

!    Declarations
!
        REAL    VP_WIDTH,VP_HEIGHT,A(100,2),B(100,2),XL2,wc_x1,wc_y1
        REAL    wc_x2,wc_y2

!    Declare global variables
!
        COMMON  wc_x1,wc_y1,wc_x2,wc_y2,vp_width,vp_height
        COMMON  new_abs_x,new_abs_y,wd_id,vd_id
!
!    Constants
!
        NUMLINES = 10                   ! number of lines drawn
        MAXLINELEN = 1024
        WID = 1024                      ! Initial sid of cube
        LEN =  860                      ! Initial length of cube
        WC_X1 = 0.                      ! World coordinate X1 of viewport
        WC_Y1 = 0.                      ! World coordinate Y1 of viewport
        WC_X2 = 1024                    ! World coordinate X2 of viewport
        WC_Y2 = 860                     ! World coordinate Y2 of viewport
C
C Prompt the user for the viewport dimensions.  Do not allow the user to
C specify viewport dimensions of less than 3/10 of a cm or greater than 70 cr
C
        PRINT *,'ENTER DESIRED WIDTH AND HEIGHT OF VIEWPORT (IN CENTIMETERS)'
        ACCEPT *,VP_WIDTH,VP_HEIGHT
        IF (vp_width  .LT. .30)  THEN
                vp_width  = .30
        ELSE IF (vp_width .GT. 70) THEN
                vp_width = 70
        END IF
        IF (vp_height .LT. .30)  THEN
                vp_height = .30
        ELSE IF (vp_height .GT. 70) THEN
                vp_height = 70
        END IF
C
C Create the display and window.  Enable the window resize option.
C
        CALL UIS$GET_HW_COLOR_INFO('SYS$WORKSTATION',,
        1                       VCM_SIZE)
        VCM_SIZE=16
        IF (VCM_SIZE .EQ. 2) GOTO 55
        VCM_SIZE = VCM_SIZE/4
        IF (VCM_SIZE .LT. NUM_LINES) GOTO 55
        NUM_LINES = 20
55      VCM_ID = UIS$CREATE_COLOR_MAP(VCM_SIZE)
        VD_ID = UIS$CREATE_DISPLAY(WC_X1,WC_Y1,WC_X2,WC_Y2,
        1                          VP_WIDTH,VP_HEIGHT,VCM_ID)
        CALL UIS$DISABLE_DISPLAY_LIST(VD_ID)
        CALL CREATE_COLORS(VCM_SIZE,VD_ID)
        WD_ID = UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','QIX')
        CALL UIS$SET_RESIZE_AST(vd_id,wd_id,ENABLE_WINDOW_RESIZE,dummy,
        1       new_abs_x,new_abs_y,vp_width,vp_height)
C
C ATTRIBUTE BLOCK 0 = WRITING MODE OVERLAY
C ATTRIBUTE BLOCK 1 = WRITING MODE ERASE
C
        CALL UIS$SET_WRITING_MODE(VD_ID, 0, 1, 9)
C
C Randomize the initial endpoints of the cube.
C
        A1 = INT(RAN(IX)*WID)
        A2 = INT(RAN(IX)*LEN)
        B1 = INT(RAN(IX)*WID)
        B2 = INT(RAN(IX)*LEN)
```

```
      CALL RANXY(A1MOD,A2MOD)
      CALL RANXY(B1MOD,B2MOD)
      ERASE = .FALSE.

      DO 66 L = 2,VCM_SIZE+2-1
            CALL UIS$SET_WRITING_INDEX(VD_ID,0,L,L-2)
66    CONTINUE

10    DO 20 L = 1,NUMLINES

          IF (ERASE) THEN
              CALL UIS$PLOT(VD_ID, 1, A(L,1),A(L,2),B(L,1),B(L,2))
              R = R + 1.0
          ENDIF

          OLDA1 = A1
          OLDA2 = A2
          OLDB1 = B1
          OLDB2 = B2

12        CALL NEWPT(A1,A1MOD,A2,A2MOD)
              L2 = (((A1+B1)/4)**2) + ((A2+B2)**2)
              XL2 = L2
              XL2 = SQRT(XL2)
              LINELEN = JIFIX(XL2)

      If (LINELEN .LE. MAXLINELEN) GOTO 15
      CALL RANXY(A1MOD,A2MOD)
      A1 = OLDA1
      A2 = OLDA2
      GOTO 12

15        CALL NEWPT(B1,B1MOD,B2,B2MOD)

              L2 = (((A1+B1)/4)**2) + ((A2+B2)**2)
              XL2 = L2
              XL2 = SQRT(XL2)
              LINELEN = JIFIX(XL2)

      If (LINELEN .LE. MAXLINELEN) GOTO 17
      CALL RANXY(B1MOD,B2MOD)
      B1 = OLDB1
      B2 = OLDB2
      GOTO 15

17    A(L,1) = A1
      A(L,2) = A2
      B(L,1) = B1
      B(L,2) = B2
      CALL UIS$PLOT(VD_ID, COUNTER, A(L,1),A(L,2),B(L,1),B(L,2))
C     TYPE *,'DRAW LINE FROM (',A1,',',A2,') TO (',B1,',',B2,')'
      COUNTER=COUNTER+1
      IF (COUNTER .LT. VCM_SIZE+2) GOTO 20
      COUNTER = 2
20    CONTINUE

      ERASE = .TRUE.

      GOTO 10

      END


      SUBROUTINE RANXY(X,Y)
C
C     This subroutine randomizes the values of the integers x and y
C     that are passed as input.
C
      INTEGER X,Y
```

```
      ISTP = 40
      X = INT(RAN(IX)*ISTP) - (ISTP/2)
      Y = ISTP/2 - ABS(X)
      IF (RAN(IX) .GT. 0.5) Y = Y * (-1)
      RETURN
      END


      SUBROUTINE enable_window_resize
C
C     This is the AST routine for the window resize.  It will not allow
C     the user to make the window dimensions less than 3/10 centimeter.
C
      COMMON  wc_x1,wc_y1,wc_x2,wc_y2,vp_width,vp_height
      COMMON  new_abs_x,new_abs_y,wd_id,vd_id

      IF (vp_width .LT. .25)  vp_width  = .25
      IF (vp_height .LT. .25) vp_height = .25
      CALL UIS$RESIZE_WINDOW(vd_id,wd_id,new_abs_x,new_abs_y,
     1       vp_width,vp_height,
     2       wc_x1,wc_y1,wc_x2,wc_y2)
      RETURN
      END


      SUBROUTINE NEWPT(X,XMOD,Y,YMOD,MAX)
C
C     This subroutine computes new values for the coordinates x and y that
C     are passed as input.  If the coordinates of the new point computed
C     are greater than 1024 (WID) or 860 (LEN), then the lines have hit the
C     walls of the window.  In such a case, the coordinates are recomputed.
C
      INTEGER X,Y,XMOD,YMOD
      INTEGER XSAV,YSAV

      XSAV = X
      YSAV = Y
50    X = X + XMOD
      IF ((X .GT. 1024) .OR. (X .LT. 0)) GOTO 100
      Y = Y + YMOD
      IF ((Y .GT. 860) .OR. (Y .LT. 0)) GOTO 100
      RETURN

100   CALL RANXY(XMOD,YMOD)
      X = XSAV
      Y = YSAV
      GOTO 50

      END
```

F.2 The Annotated FORTRAN Program

This section shows the annotated program:

```
        PROGRAM QIX
c ************************************************************************
c *                                                                      *
c *  COPYRIGHT © 1983, 1985, 1986, 1987 BY                               *
c *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTTS.             *
c *  ALL RIGHTS RESERVED.                                                *
c *                                                                      *
c *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND      *
c *  COPIED ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE      *
c *  AND WITH  THE  INCLUSION  OF  THE ABOVE COPYRIGHT NOTICE. THIS      *
c *  SOFTWARE OR  ANY  OTHER COPIES  THEREOF MAY NOT BE PROVIDED OR      *
c *  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON.  NO TITLE TO AND      *
c *  OWNERSHIP OF  THE  SOFTWARE IS  HEREBY TRANSFERRED.                 *
c *                                                                      *
c *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT      *
c *  NOTICE  AND  SHOULD  NOT  BE  CONSTRUED  AS  A  COMMITMENT  BY      *
c *  DIGITAL EQUIPMENTCORPORATION.                                       *
c *                                                                      *
c *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY      *
c *  OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.      *
c *                                                                      *
c *                                                                      *
c ************************************************************************

!++
! FACILITY:
!    Qix demo for VaxStation II
!
! ABSTRACT:
!   This program is an animation demo that moves a bunch of lines around
!   within the window.  There are 10 lines.  The line currently at the front
!   of the bunch is repeatedly erased then redrawn at the rear of the bunch
!   to create the illusion that the lines are moving.  When the lines
!   hit any side of the window, they are deflected off.
!
!
!--
!    Implicit inputs
!
       IMPLICIT INTEGER(A-Z)

!    Include files
!
       INCLUDE 'SYS$LIBRARY:UISU$RDEF'
       INCLUDE 'SYS$LIBRARY:UISENTRY'

!    Declare AST routines as external
!
       EXTERNAL enable_window_resize

!    Declarations
!
       REAL    VP_WIDTH,VP_HEIGHT,A(100,2),B(100,2),XL2,wc_x1,wc_y1
       REAL    wc_x2,wc_y2
```

Sample FORTRAN Program (QIX.FOR)

```
!    Declare global variables
!
        COMMON  wc_x1,wc_y1,wc_x2,wc_y2,vp_width,vp_height
        COMMON  new_abs_x,new_abs_y,wd_id,vd_id
!
!    Constants
!
        NUMLINES = 10                   ! number of lines drawn
        MAXLINELEN = 1024
        WID = 1024                      ! Initial wid of cube
        LEN =  860                      ! Initial length of cube
        WC_X1 = 0.                      ! World coordinate X1 of viewport
        WC_Y1 = 0.                      ! World coordinate Y1 of viewport
        WC_X2 = 1024                    ! World coordinate X2 of viewport
        WC_Y2 = 860                     ! World coordinate Y2 of viewport
C
C Prompt the user for the viewport dimensions.  Do not allow the user to
C specify viewport dimensions of less than 3/10 of a cm or greater than 70 c
C
        PRINT *,'ENTER DESIRED WIDTH AND HEIGHT OF VIEWPORT (IN CENTIMETERS)
        ACCEPT *,VP_WIDTH,VP_HEIGHT
        IF (vp_width  .LT. .30)  THEN
                vp_width  = .30
        ELSE IF (vp_width .GT. 70) THEN
                vp_width = 70
        END IF
        IF (vp_height .LT. .30)  THEN
                vp_height = .30
        ELSE IF (vp_height .GT. 70) THEN
                vp_height = 70
        END IF
C
C Create the display and window.  Enable the window resize option.
C
C    %UIS%  Information is available through a number of individual
C    calls - Please see "Display Routines".
       CALL UIS$GET_HW_COLOR_INFO('SYS$WORKSTATION',,
        1                         VCM_SIZE)
        VCM_SIZE=16
        IF (VCM_SIZE .EQ. 2) GOTO 55
        VCM_SIZE = VCM_SIZE/4
        IF (VCM_SIZE .LT. NUM_LINES) GOTO 55
        NUM_LINES = 20
C   %UIS%  Color maps may be created by using the X$ALLOC_COLOR_CELLS.
55      VCM_ID = UIS$CREATE_COLOR_MAP(VCM_SIZE)
C   %UIS%  No equivalent routine exists.
        VD_ID = UIS$CREATE_DISPLAY(WC_X1,WC_Y1,WC_X2,WC_Y2,
        1                          VP_WIDTH,VP_HEIGHT,VCM_ID)
C   %UIS%  X11 provides no equivalents to the UIS$ display list routines.
        CALL UIS$DISABLE_DISPLAY_LIST(VD_ID)
        CALL CREATE_COLORS(VCM_SIZE,VD_ID)
C   %UIS%  Please see information on virtual displays.
        WD_ID = UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION','QIX')
C   %UIS%  There are equivalent X events.
        CALL UIS$SET_RESIZE_AST(vd_id,wd_id,ENABLE_WINDOW_RESIZE,dummy,
        1       new_abs_x,new_abs_y,vp_width,vp_height)
C
C ATTRIBUTE BLOCK 0 = WRITING MODE OVERLAY
C ATTRIBUTE BLOCK 1 = WRITING MODE ERASE
C
C   %UIS%  UIS$WRITING_MODE is similar to X$SET_FUNCTION or X$CHANGE_GC.
        CALL UIS$SET_WRITING_MODE(VD_ID, 0, 1, 9)
C
C Randomize the initial endpoints of the cube.
C
        A1 = INT(RAN(IX)*WID)
        A2 = INT(RAN(IX)*LEN)
        B1 = INT(RAN(IX)*WID)
        B2 = INT(RAN(IX)*LEN)
```

```
        CALL RANXY(A1MOD,A2MOD)
        CALL RANXY(B1MOD,B2MOD)
        ERASE = .FALSE.

        DO 66 L = 2,VCM_SIZE+2-1
C   %UIS%  UIS$WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_GC.
              CALL UIS$SET_WRITING_INDEX(VD_ID,0,L,L-2)
66      CONTINUE

10      DO 20 L = 1,NUMLINES

            IF (ERASE) THEN
C   %UIS%  UIS$PLOT is similar to X$DRAW_LINE, X$DRAW_LINES or X$DRAW_POINT.

                CALL UIS$PLOT(VD_ID, 1, A(L,1),A(L,2),B(L,1),B(L,2))
                R = R + 1.0
            ENDIF

            OLDA1 = A1
            OLDA2 = A2
            OLDB1 = B1
            OLDB2 = B2

12          CALL NEWPT(A1,A1MOD,A2,A2MOD)
                L2 = (((A1+B1)/4)**2) + ((A2+B2)**2)
                XL2 = L2
                XL2 = SQRT(XL2)
                LINELEN = JIFIX(XL2)

        If (LINELEN .LE. MAXLINELEN) GOTO 15
        CALL RANXY(A1MOD,A2MOD)
        A1 = OLDA1
        A2 = OLDA2
        GOTO 12

15          CALL NEWPT(B1,B1MOD,B2,B2MOD)

                L2 = (((A1+B1)/4)**2) + ((A2+B2)**2)
                XL2 = L2
                XL2 = SQRT(XL2)
                LINELEN = JIFIX(XL2)

        If (LINELEN .LE. MAXLINELEN) GOTO 17
        CALL RANXY(B1MOD,B2MOD)
        B1 = OLDB1
        B2 = OLDB2
        GOTO 15

17      A(L,1) = A1
        A(L,2) = A2
        B(L,1) = B1
        B(L,2) = B2
C   %UIS%  UIS$PLOT is similar to X$DRAW_LINE, X$DRAW_LINES or X$DRAW_POINT.

        CALL UIS$PLOT(VD_ID, COUNTER, A(L,1),A(L,2),B(L,1),B(L,2))
C       TYPE *,'DRAW LINE FROM (',A1,',',A2,') TO (',B1,',',B2,')'
        COUNTER=COUNTER+1
        IF (COUNTER .LT. VCM_SIZE+2) GOTO 20
        COUNTER = 2
20      CONTINUE

        ERASE = .TRUE.

        GOTO 10

        END

        SUBROUTINE RANXY(X,Y)
C
C       This subroutine randomizes the values of the integers x and y that
C       are passed as input.
C
        INTEGER X,Y
```

```
        ISTP = 40
        X = INT(RAN(IX)*ISTP) - (ISTP/2)
        Y = ISTP/2 - ABS(X)
        IF (RAN(IX) .GT. 0.5) Y = Y * (-1)
        RETURN
        END


        SUBROUTINE enable_window_resize
C
C       This is the AST routine for the window resize.  It will not allow th
C       user to make the window dimensions less than 3/10 centimeter.
C
        COMMON  wc_x1,wc_y1,wc_x2,wc_y2,vp_width,vp_height
        COMMON  new_abs_x,new_abs_y,wd_id,vd_id

        IF (vp_width .LT. .25)  vp_width  = .25
        IF (vp_height .LT. .25) vp_height = .25
C   %UIS%  UIS$RESIZE_WINDOW is similar to X$CHANGE_WINDOW_ATTRIBUTES.
        CALL UIS$RESIZE_WINDOW(vd_id,wd_id,new_abs_x,new_abs_y,
     1       vp_width,vp_height,
     2       wc_x1,wc_y1,wc_x2,wc_y2)
        RETURN
        END


        SUBROUTINE NEWPT(X,XMOD,Y,YMOD,MAX)
C
C       This subroutine computes new values for the coordinates x and y that
C       are passed as input.  If the coordinates of the new point computed
C       are greater than 1024 (WID) or 860 (LEN), then the lines have hit th
C       walls of the window.  In such a case, the coordinates are recomputed
C
        INTEGER X,Y,XMOD,YMOD
        INTEGER XSAV,YSAV

        XSAV = X
        YSAV = Y
50      X = X + XMOD
        IF ((X .GT. 1024) .OR. (X .LT. 0)) GOTO 100
        Y = Y + YMOD
        IF ((Y .GT. 860) .OR. (Y .LT. 0)) GOTO 100
        RETURN

100     CALL RANXY(XMOD,YMOD)
        X = XSAV
        Y = YSAV
        GOTO 50

        END
```

F.3 The Summary Report

This section shows the Summary Report the Annotator produced for the FORTRAN program it annotated:

```
        QIX.LOG
        Date : 22-NOV-89,  Time : 08:51:02

        This report is the result of a simple search of the following files
        searching for UIS$xxx calls within programs. A summary will
        appear at the end of this report.

    >>> Examining : DISK5:[WINGATE.UISANN.SRC]QIX.FOR
=== Creating  : DISK5:[WINGATE.UISANN.SRC]QIX.FOR

        Found:          1 - UIS$CREATE_COLOR_MAP
        Color maps may be created by using the X$ALLOC_COLOR_CELLS.

        Found:          1 - UIS$CREATE_DISPLAY
        No equivalent routine exists.

        Found:          1 - UIS$CREATE_WINDOW
        Please see information on virtual displays.

        Found:          1 - UIS$DISABLE_DISPLAY_LIST
        X11 provides no equivalents to the UIS$ display list routines.

        Found:          1 - UIS$GET_HW_COLOR_INFO
        Information is available through a number of individual calls
        - Please see "Display Routines".

        Found:          2 - UIS$PLOT
        UIS$PLOT is similar to X$DRAW_LINE, X$DRAW_LINES or X$DRAW_POINT.

        Found:          1 - UIS$RESIZE_WINDOW
        UIS$RESIZE_WINDOW is similar to X$CHANGE_WINDOW_ATTRIBUTES.

        Found:          1 - UIS$SET_RESIZE_AST
        There are equivalent X events.

        Found:          1 - UIS$SET_WRITING_INDEX
        UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_GC.

        Found:          1 - UIS$SET_WRITING_MODE
        UIS$SET_WRITING_MODE is similar to X$SET_FUNCTION or X$CHANGE_GC.

       * Total Lines read in :         245

       * Total UIS calls (of any type) detected :          11

    *** Summary Information ----------------------------------------------

       * Total UIS calls (of any type) - all files :          11
```

G Sample Pascal Program (UISDC_HOUSE.PAS)

This appendix contains two versions of a sample Pascal program and a summary report. The first version is the original program before the user ran it through the Annotator. The second version is the annotated program. The Annotator also produced the summary report for the program it annotated.

G.1 The Original Pascal Program

This section shows the original Pascal program before the user ran it through the Annotator:

```
[INHERIT ('SYS$LIBRARY:UISENTRY.PEN','SYS$LIBRARY:UISUSRDEF.PEN',
          'SYS$LIBRARY:STARLET.PEN')]
{
{                        COPYRIGHT © 1989 BY
{              DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
{
{ THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
{ ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
{ INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
{ COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
{ OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
{ TRANSFERRED.
{
{ THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
{ AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
{ CORPORATION.
{
{ DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
{ SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
{ }

PROGRAM HOUSE(INPUT, OUTPUT);

VAR
 VCM_ID, WD_ID, VD_ID : UNSIGNED;
 RETWIDTH,RETHEIGHT : REAL;
 RETRESOLX,RETRESOLY : REAL;
 RETPWIDTH, RETPHEIGHT : INTEGER;
 MAX_X, MAX_Y : REAL;
 CHAR_WIDTH, CHAR_HEIGHT : INTEGER;
 I : CHAR;
 SCALING : BOOLEAN;
 RED_COLORS, GREEN_COLORS, BLUE_COLORS : ARRAY [1..6] OF REAL;
 X_ARRAY, Y_ARRAY : ARRAY [1..20] OF INTEGER;

BEGIN
 UIS$GET_DISPLAY_SIZE ('SYS$WORKSTATION',RETWIDTH,RETHEIGHT,RETRESOLX,
        RETRESOLY, RETPWIDTH, RETPHEIGHT);

 MAX_X := 22.0 * RETRESOLX;
 MAX_Y := 22.0 * RETRESOLY;

 VCM_ID := UIS$CREATE_COLOR_MAP(6);
 VD_ID :=  UIS$CREATE_DISPLAY(0.0,0.0,MAX_X,  MAX_Y,  RETPWIDTH,  RETPHEIGHT,
                              VCM_ID);
 RED_COLORS[1] := 1.0;
 GREEN_COLORS[1] := 1.0;
 BLUE_COLORS[1] := 1.0;
```

```
RED_COLORS[2] := 0.0;
GREEN_COLORS[2] := 0.0;
BLUE_COLORS[2] := 0.0;

RED_COLORS[3] := 0.0;
GREEN_COLORS[3] := 1.0;
BLUE_COLORS[3] := 0.0;

RED_COLORS[4] := 1.0;
GREEN_COLORS[4] := 0.0;
BLUE_COLORS[4] := 0.0;

RED_COLORS[5] := 1.0;
GREEN_COLORS[5] := 1.0;
BLUE_COLORS[5] := 0.0;

RED_COLORS[6] := 0.0;
GREEN_COLORS[6] := 0.0;
BLUE_COLORS[6] := 0.0;

UIS$SET_COLORS(VD_ID, 0, 6, RED_COLORS, GREEN_COLORS, BLUE_COLORS);

WD_ID := UIS$CREATE_WINDOW(VD_ID, 'SYS$WORKSTATION','Have A Nice Day');

UIS$SET_WRITING_INDEX(VD_ID, 0, 1, 2);
UIS$SET_WRITING_INDEX(VD_ID, 0, 2, 3);
UIS$SET_WRITING_INDEX(VD_ID, 0, 3, 4);
UIS$SET_WRITING_INDEX(VD_ID, 0, 4, 5);

UIS$SET_FONT(VD_ID, 1, 1, 'UIS$FILL_PATTERNS');
UIS$SET_FILL_PATTERN(VD_ID, 1, 1, PATT$C_FOREGROUND);
UIS$SET_FONT(VD_ID, 2, 2, 'UIS$FILL_PATTERNS');
UIS$SET_FILL_PATTERN(VD_ID, 2, 2, PATT$C_FOREGROUND);
UIS$SET_FONT(VD_ID, 3, 3, 'UIS$FILL_PATTERNS');
UIS$SET_FILL_PATTERN(VD_ID, 3, 3, PATT$C_FOREGROUND);
UIS$SET_FONT(VD_ID, 4, 4, 'UIS$FILL_PATTERNS');
UIS$SET_FILL_PATTERN(VD_ID, 4, 4, PATT$C_FOREGROUND);

UISDC$PLOT(WD_ID, 1, 0, (RETPHEIGHT DIV 3), RETPWIDTH, (RETPHEIGHT DIV 3),
           RETPWIDTH, 0, 0, 0);

X_ARRAY[1] := RETPWIDTH DIV 4;
X_ARRAY[2] := X_ARRAY[1];
X_ARRAY[3] := RETPWIDTH DIV 2;
X_ARRAY[4] := X_ARRAY[3];

Y_ARRAY[1] := RETPHEIGHT DIV 3;
Y_ARRAY[2] := Y_ARRAY[1] + (RETPWIDTH DIV 4);
Y_ARRAY[3] := Y_ARRAY[2];
Y_ARRAY[4] := Y_ARRAY[1];

UISDC$PLOT_ARRAY(WD_ID, 2, 4, X_ARRAY, Y_ARRAY);

UISDC$PLOT(WD_ID, 4, X_ARRAY[2], Y_ARRAY[2], ((3*RETPWIDTH) DIV 8),
           Y_ARRAY[2]+(RETPHEIGHT DIV 7), X_ARRAY[3], Y_ARRAY[3]);

UISDC$PLOT(WD_ID, 4, ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) - 20, Y_ARRAY[1],
           ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) - 20, Y_ARRAY[1] + 80,
           ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) + 20, Y_ARRAY[1] + 80,
           ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) + 20, Y_ARRAY[1]);

UISDC$CIRCLE(WD_ID, 3, RETPWIDTH-150, RETPHEIGHT-150, 50);

X_ARRAY[1] := RETPWIDTH-100;
X_ARRAY[2] := RETPWIDTH-50;
X_ARRAY[3] := RETPWIDTH-250;
X_ARRAY[4] := RETPWIDTH-200;
X_ARRAY[5] := RETPWIDTH-150;
X_ARRAY[6] := RETPWIDTH-150;
X_ARRAY[7] := RETPWIDTH-150;
X_ARRAY[8] := RETPWIDTH-150;
```

```
 Y_ARRAY[1] := RETPHEIGHT-150;
 Y_ARRAY[2] := RETPHEIGHT-150;
 Y_ARRAY[3] := RETPHEIGHT-150;
 Y_ARRAY[4] := RETPHEIGHT-150;
 Y_ARRAY[5] := RETPHEIGHT-100;
 Y_ARRAY[6] := RETPHEIGHT-50;
 Y_ARRAY[7] := RETPHEIGHT-200;
 Y_ARRAY[8] := RETPHEIGHT-250;

 UISDC$LINE_ARRAY(WD_ID, 3, 8, X_ARRAY, Y_ARRAY);

 UISDC$SET_CHAR_SIZE(WD_ID, 0, 6, 'G', 15, 20);

 UISDC$TEXT(WD_ID, 6, 'Have a Nice Day!', 50, RETPHEIGHT-50);

 READLN (INPUT, I);

END.
```

G.2 The Annotated Pascal Program

This section shows the annotated program:

```
[INHERIT ('SYS$LIBRARY:UISENTRY.PEN','SYS$LIBRARY:UISUSRDEF.PEN',
      'SYS$LIBRARY:STARLET.PEN')]
{
{                          COPYRIGHT © 1989 BY
{             DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
{
{ THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
{ ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
{ INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
{ COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
{ OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
{ TRANSFERRED.
{
{ THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
{ AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
{ CORPORATION.
{
{ DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
{ SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
{ }

PROGRAM HOUSE(INPUT, OUTPUT);

VAR
 VCM_ID, WD_ID, VD_ID : UNSIGNED;
 RETWIDTH,RETHEIGHT : REAL;
 RETRESOLX,RETRESOLY : REAL;
 RETPWIDTH, RETPHEIGHT : INTEGER;
 MAX_X, MAX_Y : REAL;
 CHAR_WIDTH, CHAR_HEIGHT : INTEGER;
 I : CHAR;
 SCALING : BOOLEAN;
 RED_COLORS, GREEN_COLORS, BLUE_COLORS : ARRAY [1..6] OF REAL;
 X_ARRAY, Y_ARRAY : ARRAY [1..20] OF INTEGER;

BEGIN
{   %UIS%  This may be emulated using X$DISPLAY_WIDTH, X$DISPLAY_WIDTHMM,
X$DISPLAY_HEIGHT, and X$DISPLAY_HEIGHTMM. }   UIS$GET_DISPLAY_SIZE
('SYS$WORKSTATION',RETWIDTH,RETHEIGHT,RETRESOLX, RETRESOLY, RETPWIDTH,
RETPHEIGHT);

 MAX_X := 22.0 * RETRESOLX;
 MAX_Y := 22.0 * RETRESOLY;

{   %UIS%  Color maps may be created by using the X$ALLOC_COLOR_CELLS. }
 VCM_ID := UIS$CREATE_COLOR_MAP(6);
{   %UIS%  No equivalent routine exists. }
 VD_ID :=  UIS$CREATE_DISPLAY(0.0,0.0,MAX_X,  MAX_Y,  RETPWIDTH,  RETPHEIGHT
          VCM_ID);
 RED_COLORS[1] := 1.0;
 GREEN_COLORS[1] := 1.0;
 BLUE_COLORS[1] := 1.0;

 RED_COLORS[2] := 0.0;
 GREEN_COLORS[2] := 0.0;
 BLUE_COLORS[2] := 0.0;

 RED_COLORS[3] := 0.0;
 GREEN_COLORS[3] := 1.0;
 BLUE_COLORS[3] := 0.0;

 RED_COLORS[4] := 1.0;
 GREEN_COLORS[4] := 0.0;
 BLUE_COLORS[4] := 0.0;
```

```
 RED_COLORS[5] := 1.0;
 GREEN_COLORS[5] := 1.0;
 BLUE_COLORS[5] := 0.0;

 RED_COLORS[6] := 0.0;
 GREEN_COLORS[6] := 0.0;
 BLUE_COLORS[6] := 0.0;

{   %UIS%  UIS$SET_COLOR is equivalent to X$STORE_COLOR. }
{   %UIS%  UIS$SET_COLORS is equivalent to X$STORE_COLORS. }
 UIS$SET_COLORS(VD_ID, 0, 6, RED_COLORS, GREEN_COLORS, BLUE_COLORS);

{   %UIS%  Please see information on virtual displays. }
 WD_ID := UIS$CREATE_WINDOW(VD_ID, 'SYS$WORKSTATION','Have A Nice Day');

{   %UIS%  UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or
X$CHANGE_GC. }
 UIS$SET_WRITING_INDEX(VD_ID, 0, 1, 2);
{   %UIS%  UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or
X$CHANGE_GC. }
 UIS$SET_WRITING_INDEX(VD_ID, 0, 2, 3);
{   %UIS%  UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or
X$CHANGE_GC. }
 UIS$SET_WRITING_INDEX(VD_ID, 0, 3, 4);
{   %UIS%  UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or
X$CHANGE_GC. }
 UIS$SET_WRITING_INDEX(VD_ID, 0, 4, 5);

{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is obtained
from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 1, 1, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC. }
 UIS$SET_FILL_PATTERN(VD_ID, 1, 1, PATT$C_FOREGROUND);
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is obtained
from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 2, 2, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC. }
 UIS$SET_FILL_PATTERN(VD_ID, 2, 2, PATT$C_FOREGROUND);
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is obtained
from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 3, 3, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC. }
 UIS$SET_FILL_PATTERN(VD_ID, 3, 3, PATT$C_FOREGROUND);
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is obtained
from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 4, 4, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC. }
 UIS$SET_FILL_PATTERN(VD_ID, 4, 4, PATT$C_FOREGROUND);

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT(WD_ID, 1, 0, (RETPHEIGHT DIV 3), RETPWIDTH, (RETPHEIGHT DIV 3),
     RETPWIDTH, 0, 0, 0);

 X_ARRAY[1] := RETPWIDTH DIV 4;
 X_ARRAY[2] := X_ARRAY[1];
 X_ARRAY[3] := RETPWIDTH DIV 2;
 X_ARRAY[4] := X_ARRAY[3];

 Y_ARRAY[1] := RETPHEIGHT DIV 3;
 Y_ARRAY[2] := Y_ARRAY[1] + (RETPWIDTH DIV 4);
 Y_ARRAY[3] := Y_ARRAY[2];
 Y_ARRAY[4] := Y_ARRAY[1];

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
{   %UIS%  UISDC$PLOT_ARRAY is similar to X$DRAW_LINE, X$DRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT_ARRAY(WD_ID, 2, 4, X_ARRAY, Y_ARRAY);
```

Sample Pascal Program (UISDC_HOUSE.PAS)

```
{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT(WD_ID, 4, X_ARRAY[2], Y_ARRAY[2], ((3*RETPWIDTH) DIV 8),
      Y_ARRAY[2]+(RETPHEIGHT DIV 7), X_ARRAY[3], Y_ARRAY[3]);

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT(WD_ID, 4, ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) - 20, Y_ARRAY[1],
      ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) - 20, Y_ARRAY[1] + 80,
     ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) + 20, Y_ARRAY[1] + 80,
     ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) + 20, Y_ARRAY[1]);

{   %UIS%  UISDC$CIRCLE is similar to X$DRAW_ARC. }
 UISDC$CIRCLE(WD_ID, 3, RETPWIDTH-150, RETPHEIGHT-150, 50);

 X_ARRAY[1] := RETPWIDTH-100;
 X_ARRAY[2] := RETPWIDTH-50;
 X_ARRAY[3] := RETPWIDTH-250;
 X_ARRAY[4] := RETPWIDTH-200;
 X_ARRAY[5] := RETPWIDTH-150;
 X_ARRAY[6] := RETPWIDTH-150;
 X_ARRAY[7] := RETPWIDTH-150;
 X_ARRAY[8] := RETPWIDTH-150;

 Y_ARRAY[1] := RETPHEIGHT-150;
 Y_ARRAY[2] := RETPHEIGHT-150;
 Y_ARRAY[3] := RETPHEIGHT-150;
 Y_ARRAY[4] := RETPHEIGHT-150;
 Y_ARRAY[5] := RETPHEIGHT-100;
 Y_ARRAY[6] := RETPHEIGHT-50;
 Y_ARRAY[7] := RETPHEIGHT-200;
 Y_ARRAY[8] := RETPHEIGHT-250;

{   %UIS%  UISDC$LINE is similar to X$DRAW_SEGMENT or X$DRAW_POINT. }
{   %UIS%  UISDC$LINE_ARRAY is similar to X$DRAW_SEGMENTS or X$DRAW_POINTS.
 UISDC$LINE_ARRAY(WD_ID, 3, 8, X_ARRAY, Y_ARRAY);

{   %UIS%  X11 does not provide text scaling. }
 UISDC$SET_CHAR_SIZE(WD_ID, 0, 6, 'G', 15, 20);

{   %UIS%  UISDC$TEXT is similar to X$DRAW_TEXT. }
 UISDC$TEXT(WD_ID, 6, 'Have a Nice Day!', 50, RETPHEIGHT-50);

 READLN (INPUT, I);

END.
```

```
[INHERIT ('SYS$LIBRARY:UISENTRY.PEN','SYS$LIBRARY:UISUSRDEF.PEN',
          'SYS$LIBRARY:STARLET.PEN')]
{
{                        COPYRIGHT © 1989 BY
{             DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
{
{ THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
{ ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
{ INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR  ANY  OTHER
{ COPIES  THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
{ OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE  IS  HEREBY
{ TRANSFERRED.
{
{ THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
{ AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
{ CORPORATION.
{
{ DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR  RELIABILITY  OF  ITS
{ SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
{ }

PROGRAM HOUSE(INPUT, OUTPUT);
```

```
VAR
 VCM_ID, WD_ID, VD_ID : UNSIGNED;
 RETWIDTH,RETHEIGHT : REAL;
 RETRESOLX,RETRESOLY : REAL;
 RETPWIDTH, RETPHEIGHT : INTEGER;
 MAX_X, MAX_Y : REAL;
 CHAR_WIDTH, CHAR_HEIGHT : INTEGER;
 I : CHAR;
 SCALING : BOOLEAN;
 RED_COLORS, GREEN_COLORS, BLUE_COLORS : ARRAY [1..6] OF REAL;
 X_ARRAY, Y_ARRAY : ARRAY [1..20] OF INTEGER;

BEGIN
{   %UIS%  This may be emulated using X$DISPLAY_WIDTH,
X$DISPLAY_WIDTHMM, X$DISPLAY_HEIGHT, and X$DISPLAY_HEIGHTMM. }
 UIS$GET_DISPLAY_SIZE ('SYS$WORKSTATION',RETWIDTH,RETHEIGHT,RETRESOLX,
        RETRESOLY, RETPWIDTH, RETPHEIGHT);

 MAX_X := 22.0 * RETRESOLX;
 MAX_Y := 22.0 * RETRESOLY;

{   %UIS%  Color maps may be created by using the X$ALLOC_COLOR_CELLS. }
 VCM_ID := UIS$CREATE_COLOR_MAP(6);
{   %UIS%  No equivalent routine exists. }
 VD_ID :=  UIS$CREATE_DISPLAY(0.0,0.0,MAX_X,  MAX_Y,  RETPWIDTH,  RETPHEIGHT,
                              VCM_ID);
 RED_COLORS[1] := 1.0;
 GREEN_COLORS[1] := 1.0;
 BLUE_COLORS[1] := 1.0;

 RED_COLORS[2] := 0.0;
 GREEN_COLORS[2] := 0.0;
 BLUE_COLORS[2] := 0.0;

 RED_COLORS[3] := 0.0;
 GREEN_COLORS[3] := 1.0;
 BLUE_COLORS[3] := 0.0;

 RED_COLORS[4] := 1.0;
 GREEN_COLORS[4] := 0.0;
 BLUE_COLORS[4] := 0.0;

 RED_COLORS[5] := 1.0;
 GREEN_COLORS[5] := 1.0;
 BLUE_COLORS[5] := 0.0;

 RED_COLORS[6] := 0.0;
 GREEN_COLORS[6] := 0.0;
 BLUE_COLORS[6] := 0.0;

{   %UIS%  UIS$SET_COLOR is equivalent to X$STORE_COLOR. }
{   %UIS%  UIS$SET_COLORS is equivalent to X$STORE_COLORS. }
 UIS$SET_COLORS(VD_ID, 0, 6, RED_COLORS, GREEN_COLORS, BLUE_COLORS);

{   %UIS%  Please see information on virtual displays. }
 WD_ID := UIS$CREATE_WINDOW(VD_ID, 'SYS$WORKSTATION','Have A Nice Day');

{   %UIS%  UIS$WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_GC}
 UIS$SET_WRITING_INDEX(VD_ID, 0, 1, 2);
{   %UIS%  UIS$WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_CG}
 UIS$SET_WRITING_INDEX(VD_ID, 0, 2, 3);
{   %UIS%  UIS$WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_CG}
 UIS$SET_WRITING_INDEX(VD_ID, 0, 3, 4);
{   %UIS%  UIS$WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_CG}
 UIS$SET_WRITING_INDEX(VD_ID, 0, 4, 5);
```

```
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is
obtained from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 1, 1, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
 UIS$SET_FILL_PATTERN(VD_ID, 1, 1, PATT$C_FOREGROUND);
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is
obtained from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 2, 2, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
 UIS$SET_FILL_PATTERN(VD_ID, 2, 2, PATT$C_FOREGROUND);
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is
obtained from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 3, 3, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
 UIS$SET_FILL_PATTERN(VD_ID, 3, 3, PATT$C_FOREGROUND);
{   %UIS%  UIS$SET_FONT is similar to X$SET_FONT. The font ID is
obtained from X$LOAD_FONT. }
 UIS$SET_FONT(VD_ID, 4, 4, 'UIS$FILL_PATTERNS');
{   %UIS%  UIS fill patterns are equivalent to STIPPLE patterns in X11; use
 UIS$SET_FILL_PATTERN(VD_ID, 4, 4, PATT$C_FOREGROUND);

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT(WD_ID, 1, 0, (RETPHEIGHT DIV 3), RETPWIDTH, (RETPHEIGHT DIV 3),
            RETPWIDTH, 0, 0, 0);

 X_ARRAY[1] := RETPWIDTH DIV 4;
 X_ARRAY[2] := X_ARRAY[1];
 X_ARRAY[3] := RETPWIDTH DIV 2;
 X_ARRAY[4] := X_ARRAY[3];

 Y_ARRAY[1] := RETPHEIGHT DIV 3;
 Y_ARRAY[2] := Y_ARRAY[1] + (RETPWIDTH DIV 4);
 Y_ARRAY[3] := Y_ARRAY[2];
 Y_ARRAY[4] := Y_ARRAY[1];

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
{   %UIS%  UISDC$PLOT_ARRAY is similar to X$DRAW_LINE, X$DRAW_LINES,
or X$DRAW_POINT. }
 UISDC$PLOT_ARRAY(WD_ID, 2, 4, X_ARRAY, Y_ARRAY);

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT(WD_ID, 4, X_ARRAY[2], Y_ARRAY[2], ((3*RETPWIDTH) DIV 8),
            Y_ARRAY[2]+(RETPHEIGHT DIV 7), X_ARRAY[3], Y_ARRAY[3]);

{   %UIS%  UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or
X$DRAW_POINT. }
 UISDC$PLOT(WD_ID, 4, ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) - 20, Y_ARRAY[1],
            ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) - 20, Y_ARRAY[1] + 80,
            ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) + 20, Y_ARRAY[1] + 80,
            ((X_ARRAY[2] + X_ARRAY[3]) DIV 2) + 20, Y_ARRAY[1]);

{   %UIS%  UISDC$CIRCLE is similar to X$DRAW_ARC. }
 UISDC$CIRCLE(WD_ID, 3, RETPWIDTH-150, RETPHEIGHT-150, 50);

 X_ARRAY[1] := RETPWIDTH-100;
 X_ARRAY[2] := RETPWIDTH-50;
 X_ARRAY[3] := RETPWIDTH-250;
 X_ARRAY[4] := RETPWIDTH-200;
 X_ARRAY[5] := RETPWIDTH-150;
 X_ARRAY[6] := RETPWIDTH-150;
 X_ARRAY[7] := RETPWIDTH-150;
 X_ARRAY[8] := RETPWIDTH-150;
```

```
 Y_ARRAY[1] := RETPHEIGHT-150;
 Y_ARRAY[2] := RETPHEIGHT-150;
 Y_ARRAY[3] := RETPHEIGHT-150;
 Y_ARRAY[4] := RETPHEIGHT-150;
 Y_ARRAY[5] := RETPHEIGHT-100;
 Y_ARRAY[6] := RETPHEIGHT-50;
 Y_ARRAY[7] := RETPHEIGHT-200;
 Y_ARRAY[8] := RETPHEIGHT-250;

{   %UIS%  UISDC$LINE is similar to X$DRAW_SEGMENT or X$DRAW_POINT. }
{   %UIS%  UISDC$LINE_ARRAY is similar to X$DRAW_SEGMENTS or X$DRAW_POINTS. }
 UISDC$LINE_ARRAY(WD_ID, 3, 8, X_ARRAY, Y_ARRAY);

{   %UIS%  X11 does not provide text scaling. }
 UISDC$SET_CHAR_SIZE(WD_ID, 0, 6, 'G', 15, 20);

{   %UIS%  UISDC$TEXT is similar to X$DRAW_TEXT. }
 UISDC$TEXT(WD_ID, 6, 'Have a Nice Day!', 50, RETPHEIGHT-50);

 READLN (INPUT, I);

END.
```

G.3 The Summary Report

This section shows the Summary Report the Annotator produced for the Pascal program it annotated:

```
UISDC_HOUSE.LOG
Date : 23-MAR-90,  Time : 16:21:34

This report is the result of searching of the following files:
    UISDC_HOUSE.PAS

searching for UIS calls within programs. A summary will
appear at the end of this report.

    >>> Examining : WORK2:[SMITH.MIG]UISDC_HOUSE.PAS
=== Creating  : WORK2:[SMITH.MIG]UISDC_HOUSE.PAS

Found:          1 - UIS$CREATE_COLOR_MAP
Color maps may be created by using the X$ALLOC_COLOR_CELLS.

Found:          1 - UIS$CREATE_DISPLAY
No equivalent routine exists.

Found:          1 - UIS$CREATE_WINDOW
Please see information on virtual displays.

Found:          1 - UIS$GET_DISPLAY_SIZE
This may be emulated using X$DISPLAY_WIDTH, X$DISPLAY_WIDTHMM,
X$DISPLAY_HEIGHT, and X$DISPLAY_HEIGH

Found:          1 - UIS$SET_COLOR
UIS$SET_COLOR is equivalent to X$STORE_COLOR.

Found:          1 - UIS$SET_COLORS
UIS$SET_COLORS is equivalent to X$STORE_COLORS.

Found:          4 - UIS$SET_FILL_PATTERN
UIS fill patterns are equivalent to STIPPLE patterns in X11; use
X$SET_STIPPLE or X$CHANGE_GC

Found:          4 - UIS$SET_FONT
UIS$SET_FONT is similar to X$SET_FONT. The font ID is obtained from
X$LOAD_FONT.

Found:          4 - UIS$SET_WRITING_INDEX
UIS$SET_WRITING_INDEX is similar to X$SET_FOREGROUND or X$CHANGE_GC

Found:          1 - UISDC$CIRCLE
UISDC$CIRCLE is similar to X$DRAW_ARC.

Found:          1 - UISDC$LINE
UISDC$LINE is similar to X$DRAW_SEGMENT or X$DRAW_POINT.

Found:          1 - UISDC$LINE_ARRAY
UISDC$LINE_ARRAY is similar to X$DRAW_SEGMENTS or X$DRAW_POINTS.

Found:          4 - UISDC$PLOT
UISDC$PLOT is similar to X$DRAW_LINE, X$SRAW_LINES, or X$DRAW_POINT

Found:          1 - UISDC$PLOT_ARRAY
UISDC$PLOT_ARRAY is similar to X$DRAW_LINE, X$DRAW_LINES, or
X$DRAW_POINT.


Found:          1 - UISDC$SET_CHAR_SIZE
X11 does not provide text scaling.

Found:          1 - UISDC$TEXT
UISDC$TEXT is similar to X$DRAW_TEXT.

* Total Lines read in :         142

* Total UIS calls (of any type) detected :          25
```

```
*** Summary Information ----------------------------------------------
   * Total UIS calls (of any type) - all files :          25
```