VAXuisx User's Guide

Order Number: AA-PC3GA-TE

Software Version: VAXuisx V1.0

Operating System: VMS V5.3 or above

You must have DECwindows installed for VAXuisx to run.

August 1990

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

Preface

The *VAXuisx User's Guide* introduces you to the VAXuisx Runtime Library for VMS. You can use VAXuisx to most run UIS applications on top of DECwindows without converting, rewriting, or relinking the applications.

Note: VMS is the *only* operating system that is supported by VAXuisx.

Structure of This Manual

This manual includes one chapter, two appendixes and a glossary of terms.

Chapter 1	Contains notes on using VAXuisx.
Appendix A	Identifies VAXuisx fonts.
Appendix B	Lists logical names.
Glossary	Defines terms that may be unfamiliar or terms used differently in VWS than in DECwindows.

Related Documents

If you are working with VWS, you should 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.

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 example applications.
- *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 Runtime Library for VMS.

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.

<X

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

1 Notes on Using VAXuisx

VAXuisx allows existing applications written for the VWS windowing system to run under the DECwindows windowing system. VAXuisx accomplishes this by providing a new run-time library that uses the Xlib program interface for drawing and input services. UIS routines are redirected to the new UISXSHR image. Therefore, you do not need to relink applications or install native VWS support.

Note: To improve performance for graphic text operations, you should install the VWS fonts. You can install the fonts by using VMSINSTAL with the VWS media supplied with the VWS V4.3 installation kit and installing only the fonts.

To render UIS files produced by running your application with VAXuisx, you should install HCUIS and the fonts supplied with your VWS V4.3 installation kit. You can install them by using VMSINSTAL and selecting the installations for HCUIS and the fonts.

You should be able to run most of your UIS applications on VMS DECwindows using VAXuisx.

Note: In order to use VAXuisx, you must be running DECwindows.

VAXuisx is only supported on VMS. However, you can display on a system running ULTRIX.

1.1 Running a UIS application from DEBUG

If your graphics window becomes occluded and then exposed when running a UIS application from DEBUG, UIS repairs window damage immediately. If the same problem occurs on VAXuisx when UISX$BACKING_STORE is set to TRUE, the damage is not repaired until the application is stepped once. (Stepping once allows an AST to be delivered.) If UISX$BACKING_STORE is set to FALSE, VAXuisx does not repair the damage at all.

1.2 SYS$WORKSTATION

A number of UIS calls require a *device name* parameter. For VWS, the device name parameter must be SYS$WORKSTATION or the name of the video device driver for the workstation. In addition, several UIS routines assume the presence of SYS$WORKSTATION.

VAXuisx views SYS$WORKSTATION as the destination string for a X11 Server. This string is specified in VMS DECwindows as either a workstation device *WSAn:* or an X11 NODE::SERVER.SCREEN string. By default, VAXuisx defines SYS$WORKSTATION to point to the logical DECW$DISPLAY. On a DECwindows workstation, DECW$DISPLAY directs output to the workstation screen by default.

You can redefine SYS$WORKSTATION to other server destinations. The logical can be nested to 11 levels of logical name translation. The final value of the translated logical must be a Workstation Device WSAn: or a valid X11 NODE::SERVER.SCREEN string. You also can create workstation devices using the SET DISPLAY command in DCL.

While other logical names, explicit destination strings, or workstation devices can be specified in the UIS routine calls, VAXuisx always opens a connection to SYS$WORKSTATION. Applications should use the device string SYS$WORKSTATION whenever possible.

The example defines a destination using the workstation device mechanism:

```
$ SET DISPLAY/CREATE/NODE=mynode::/SCREEN=0/TRANSPORT=DECNET
$ SHOW DISPLAY

      Device:    WSA7:
      Node:      MYNODE
      Transport: DECNET
      Server:    0
      Screen:    0

$ DEFINE SYS$WORKSTATION WSA7:

Define a destination using X11 strings, and redirected through
multiple logical names:

$ DEFINE ANOTHER_LOGICAL mynode::0.0
$ DEFINE A_LOGICAL ANOTHER_LOGICAL
$ DEFINE SYS$WORKSTATION A_LOGICAL

This is the default VAXuisx assignment done in UISX$STARTUP.COM:

$ DEFINE SYS$WORKSTATION DECW$DISPLAY
```

Note: The default for SYS$WORKSTATION is set in the UISX$STARTUP command file and is DECW$DISPLAY.

Note: In order to run VAXuisx processes detached, both SYS$WORKSTATION and DECW$DISPLAY should be defined in executive mode (/EXECUTIVE).

The logical name search for SYS$WORKSTATION follows the normal VMS search path. Therefore, a process logical can override the system default without the user needing privileges.

1.3 Backing Store

The X Window System™ (X11) does not guarantee the integrity of the pixel data in a window. VWS does guarantee it using a feature called *backing store*. VAXuisx emulates backing store by maintaining an off-screen shadow *PIXMAP* that is the same size as the virtual display. VAXuisx uses the offscreen PIXMAP as an output buffer as well as *bit map* backup and does all writes to the PIXMAP and copies the changed areas from the PIXMAP into the window. When a window occludes part of another window, VAXuisx refreshes the destroyed area by using a copy operation from the PIXMAP to the window when that area is exposed.

X11 does not guarantee PIXMAP availability. If the PIXMAP intended for backing store is not allocated, VAXuisx will signal a fatal error. However, VAXuisx can be setup to signal a nonfatal error and continue operation. In this case, a window exposure will fill the occluded area with the background color.

Providing backing store is expensive in server resources and communications bandwidth. To save on those costs, you can disable backing store by using the logical UISX$BACKING_STORE.

1.4 Colormaps

UIS and X11 bind colormaps to the hardware differently.

1.4.1 Colormap Allocation

UIS binds all colormaps to the hardware at an offset that is an integral multiple of the size of the virtual colormap. (The colormap size is rounded up to a power of 2.) This method binds the first index in the virtual colormap to a hardware map location where all the low-order bits in the index are zero. Consequently, all the pixel values the programmer uses have the high-order bits masked off and *ORed* with the base offset into the hardware color map. This binding method is convenient for image data, which is generally stored in a *device-independent* format. It also ensures that the pixel value is within the user's virtual colormap limits after UIS performs the *ones-complement* of the low-order bits.

X11 provides a number of color models and several ways of allocating colors. Most of the ways allocate arbitrary noncontiguous index values.

VAXuisx emulates UIS virtual colormaps by using X11 routines. The method used to emulate the colormap depends on the system in use: pseudo-color, monochrome (bitonal), true color, or direct color.

Note: Pseudo-color systems include intensity systems.

On pseudo-color systems, VAXuisx emulates a colormap by using the X11 X$ALLOC_COLOR_CELLS routine to create an indirect array of pixel values. It requests one color and *num_planes* planes to allocate the smallest number of colors that entirely contains the virtual colormap. VAXuisx then permutates the plane bits and the pixel values that are returned into an array of pixel values. For all read and write image functions, VAXuisx translates all the pixel values into the correct hardware pixel values.

Note: X11 uses the formula num_colors*2^{num_planes} to allocate the smallest number of colors that entirely contains the virtual color map. When VAXuisx requests 1 Color and $^{num\_planes}$, 1 Color reduces $^{num\_planes}$ to (2^{num_planes}).

X11 does not guarantee colormap allocation.

VAXuisx uses a different method of creating a colormap if the colormap allocation fails or if the application uses colormap segments via the UIS$CREATE_COLOR_MAP_SEGMENT routine. In both cases, VAXuisx creates and uses an X11 private colormap.

Note: If VAXuisx creates a private X11 colormap, the colors already in the display will probably change. The change only happens when the VAXuisx window has keyboard focus.

On monochrome (bitonal) systems, VAXuisx creates a colormap by maintaining an array of pixel values. The method used is similar to the one it used for pseudo color systems. The pixel values it uses are those for black-and-white pixels. It sets the values using the NTSC color convention:

INTENSITY=((RED*0.30)+(GREEN*0.59)+(BLUE*0.11))

True-color systems feature a static colormap of 2^{24} colors. The 24-bit range is divided into eight bits each of red, green, and blue. VAXuisx uses its pseudo color model to indirectly map 2^{15} colors. It maps them by storing the RGB values passed to it directly. Logical operations cannot work as expected because VAXuisx is operating on the *RGB* value. Therefore, a XOR of %xFFFFFFFF on red results in cyan regardless of the index.

The VAXstation 3520/40 features a direct-colormap. You can select this map by defining the logical:

UISX$DIRECT_MAP TRUE

The direct colormap works the same as the pseudo-colormap for VAXuisx, except it allows up to 2^{15} colors instead of 256 (2^8).

1.4.2 Colormap Allocation Differences

Because DECwindows/X11 uses a different strategy to allocate colors from the hardware colormap than VWS, colormap allocation failure may occur with fewer colors than under VWS. VAXuisx must allocate the entire hardware map to satisfy a colormap request that cannot be filled from the available colors in the DECwindows default colormap. This will cause colors in other windows to change while the application window has input focus. On typical 8-plane workstations, colormaps of over 64 colors can cause this effect.

1.4.3 Colormap Segments

Colormap segments which use exact placement require the allocation of a private X11 colormap. Regardless of the size of the color map being allocated, any use of colormap segments will cause the default workstation colors of the DECwindows screen to be modified and all windows on the screen to change colors.

1.5 User Preferred Color Setup

VAXuisx will read the user's preferred color setup file DECW$SM_COLOR.DAT from the directory pointed to by the logical DECW$USER_DEFAULTS. This should make your VAXuisx windows have the same foreground/background as your default DECterm windows.

The one requirement for this feature is that the file to be read reside in the directory on the DECwindows *CLIENT*. If you run your applications under VAXuisx from a non-workstation CLIENT, you must place a copy of your DECW$SM_COLOR.DAT in the DECW$USER_DEFAULTS directory on the non-workstation system. If your file is not in the directory, you will get a default of white window background with black foreground.

The same requirement holds true for the mouse cursor. When the cursor pattern is changed, it will use the values from the DECW$SM_COLOR.DAT file to fill in the colors of the cursor, but if at application startup time it could not read the DECW$SM_COLOR.DAT file, you will get the default of white background and black foreground. Depending on how you define your cursor and its cursor mask this could make the cursor appear to be invisible in your window.

If you don't want to read the DECW$SM_COLOR.DAT file you may define the logical UISX$CUSTOMIZE_COLORS to be false.

1.6 Drawing Operations

UIS stores drawing attributes in internal structures called *Attribute Blocks (ATBs)*. For information about ATBs, refer to *VMS Workstation Software Graphics Programming Guide*.

1.6.1 Writing Modes

VAXuisx supports all the UIS writing modes. The VAXuisx writing modes are all combinations of the following X11 features:

- Function
- Fill style
- Background pixel
- Foreground pixel
- Drawing operation

VAXuisx's device-dependent modes may not provide identical results to the native UIS device-dependent modes. Those modes are:

- BIS
- BISN
- BIC
- BICN
- XOR
- COPY
- COPYN

1.6.2 Patterns

The standard UIS patterns are built into VAXuisx.

1.6.3 Line patterns

In the VWS PLOT operation (joined vectors, drawing using UIS$PLOT, UIS$PLOT_ARRAY, UISDC$PLOT, or UISDC$PLOT_ARRAY), at each line endpoint the line pattern is restarted. This is difficult to reproduce with reasonable performance characteristics and precludes the ability to draw patterned curves. In VAXuisx line patterns continue across each vector endpoint.

A side effect of this change is that patterned rubberband lines (XOR or COMPLEMENT lines) which use the PLOT operation instead of the LINE (line segments) operation will seem to shift the positions of the dashes.

1.7 Input

UIS provides input through Asynchronous System Trap (AST) routines to the application. X11 provides input queuing events that are decoded by a dispatch loop in the application. X11 can deliver a "doorbell" AST when a particular event type occurs. This feature is used by VAXuisx. When an event occurs, the AST invokes VAXuisx, which checks the event and takes the proper action.

1.7.1 Reporting Mouse Movement

In UIS, the application can request the current position of the mouse. UIS does not send all mouse movements (or events) that occurred between the last reported position and the current position. X11 sends all mouse movements to the application. VAXuisx approximates how UIS reports mouse movement. When a mouse movement occurs, VAXuisx scans the X11 *event queue* for the last mouse movement event. It stores the position the mouse stopped at as the current position and discards any earlier positions. Then, if the application requests it, VAXuisx generates a pointer movement AST.

1.7.2 Tablet/Digitizer

UIS can map the tablet input to a window. UIS confines the cursor to the window and the application controls the resolution of the input. X11 cannot map tablet input to a window. Therefore, VAXuisx does not map tablet input to a window. It always makes the tablet appear as a mouse to the UIS application.

Note: The tablet works only as a mouse because X11 does not support the tablet. None of the tablet functions work under VAXuisx.

1.8 Output Primitives

UIS output primitives are:

- Lines (Vector Drawings)
- Polygons
- Ellipses
- Text
- Images
- Scrolling logic of Move Area and Move Window

UIS and VAXuisx handle output primitives differently.

1.8.1 Lines (Vector Drawings)

UIS and X11 line attributes are similar. Two exceptions exist. The first exception is that X11 has attributes for *join style* and *cap style* while the UIS does not. (Join style refers to how two wide lines come together; cap style refers to how the end of a wide line is treated.) UIS leaves lines joined in ragged fashion that the user must fix. VAXuisx joins lines evenly, using *Miter* as its join style. In Miter style, the outer edges of the two wide lines extend until they meet at a point. VAXuisx also uses *Butt* as its cap style. With Butt style, the line is square at the endpoint, perpendicular to the slope of the line, and does not project beyond the endpoint.

The second exception is with the line width of 1. VAXuisx uses the algorithm for the X11 special-case line width of 0. X11 specifies that when line width is 0, it will use whatever algorithm the hardware does fastest. This difference between algorithms may cause lines to be off by a pixel or two in some places.

1.8.2 Polygons

UIS and X11 fill polygons differently. UIS fills a polygon inclusive of all borders. X11 fills a rectangle inclusive of only the left and the upper borders and does not draw the right and lower borders. For rectangles, VAXuisx extends the width and height of the rectangle by one pixel to draw the true border.

1.8.3 Ellipses

UIS uses two routines to draw ellipses: UIS$CIRCLE and UIS$ELLIPSE.

UIS and X11 differ in how accurately they draw circles. UIS draws to an accuracy of seven decimal places because it requires all angles to be specified as a longword floating point value. X11 draws to an accuracy of just less than two decimal places because it takes an angle as an integer value equal to the desired angle multiplied by 64. This may result in end points and edges for large arcs (radii) not being in exactly the same place as they are in UIS.

1.8.4 Text and Fonts

VAXuisx provides the standard UIS fonts in DECwindows format.

VAXuisx outputs normal text: unrotated, unscaled, and unsheared, using the standard Xlib text operations.

You should have font files available in both X11 and VWS formats. VAXuisx first tries to load the VWS font into CLIENT memory and the X11 font into *Server* memory. If the VWS font is not available, VAXuisx uses the X11 font as a template for a "fake" VWS font. If the X11 font is not available, VAXuisx draws all text as *graphic text* using the VWS font bit maps. If neither font is available, VAXuisx signals an error and uses the default font.

VAXuisx allows both VWS-style font names and DECwindows-style font names. You can "redefine" fonts using the same mechanism as VWS. To "redefine" a font, assign a logical name to the font name. You can redefine VWS font names to use X11 native fonts. The standard X11 font names, however, are illegal logical names. Using them terminates the translation of the logical name. You can nest up to ten levels of font logical names. You can also define the font logical names in the normal RMS logical name table search path.

VAXuisx provides graphic text (rotated, sheared, or scaled) by manipulating a local bit map of the text to be output and then drawing the bit map to the screen. VAXuisx obtains the bit map information for a font by first trying to load the VWS monochrome font (.VWS$FONT) from SYS$FONT. If the monochrome font is not available, VAXuisx then tries to load the VWS color font (.VWS$VAFONT). If no VWS color font is available, VAXuisx reads the individual *glyphs* from the primary workstation (SYS$WORKSTATION) through X11.

1.8.5 Images

When VAXuisx performs an Image operation, it may need to translate each pixel byte in a multibit image from the index value into a final X11 pixel value. It uses the pixel values generated when the virtual color map was created.

Read Image Operations may require that all pixels be translated from X11 color index values to the appropriate UIS color index value.

Note: Using *backing store* may produce a significant impact on application performance and CPU consumption.

1.8.5.1 UISDC$READ_IMAGE

VAXuisx implements UISDC$READ_IMAGE by first trying to get the data from the window backing store, if it exists, and then from the window. This implementation has two restrictions. The first occurs if you do not have backing store enabled (data must come from the window) and any portion of the window is damaged or occluded. The pixel values UISDC$READ_IMAGE returns will also reflect all damage or occlusion currently existing for the window. The second restriction occurs when using UISDC$READ_IMAGE on an invisible window that does not have backing store enabled. In this situation, you will not get any valid pixel data returned. All pixels will return a value of 0 (zero). Both of these restrictions are caused by restrictions in X11.

Because of these restrictions, you should always have backing store enabled (the default setting) any time you use UISDC$READ_IMAGE.

1.8.6 Scrolling

UIS performs scrolling operations by using the UIS$MOVE_AREA and the UIS$MOVE_WINDOW commands. X11 performs scrolling operations by using the X$COPY_AREA command. When VAXuisx runs on VSII monochrome workstations in which DECwindows does not use the scanline map, scrolling will be significantly slower then with VWS running on the same configuration.

1.8.7 UIS$MOVE_AREA With No Backing Store

If backing store is disabled, UIS$MOVE_AREA will copy within the window rather than copy from a PIXMAP to the window. If there are any obscured areas in the portion of the window being moved, the areas will be filled with the background color.

1.8.8 UIS$SET_POINTER_PATTERN

With VAXuisx, you cannot bind the cursor pattern to a region. Consequently, using the flag UIS$M_BIND_POINTER does not effect where the cursor can go on the display screen.

1.9 Window Management and User Interface

Window management and the user interface are different in VAXuisx and in X11. This section describes the differences.

1.9.1 Handling Icons

UIS and DECwindows handle icons differently. With UIS, the icon is visible only when in the icon state. UIS does not display the icon when the application is active. The user can position the icon anywhere on the screen.

DECwindows defines the icon by providing a PIXMAP and title to the DECwindows window manager, which then performs the icon management. The window manager always displays the icon. It grays out the icon when the application window is visible. The DECwindows window manager maintains an icon box into which it places all icons.

UISX uses the default DECwindows icon behavior. The DECwindows icon works without any problems if a UIS application uses the default icon action. But the icon may behave strangely if the UIS application uses any of the special facilities for icons.

1.9.2 Window Placement

The full window placement logic from VWS has not yet been fully implemented. The VAXuisx logic will not go out and find the most free space on the window like VWS does. The following is a brief synopsis of what VAXuisx does.

For exact placement *abs_x, abs_y*, VAXuisx basically follows the VWS logic. The only difference being that if absolute position is specified along with any of the bits for relative placement, VAXuisx ignores the relative placement bits entirely.

The bits for relative placement: top, bottom, center, left, and right, work such that a window will be placed 48 pixels from the left or right if so specified and 80 pixels from the top or bottom if so specified. Otherwise, VAXuisx will try to center the viewport between the edges. If no placement is specified, top and center will be the placement attributes given to the window.

The difference here, as opposed to VWS, is that VAXuisx does not evaluate the workstation screen for optimal placement. If an area specified currently has a viewport visible, VAXuisx will attempt to place the window in a south easterly fashion as DECwindows default placement does. If the window can fit on the screen at that point, it is placed there. If it cannot be placed there VAXuisx will move the window either left or up depending on which portion of the viewport would have been offscreen by using the down and to the right approach.

1.9.3 Window Menu Options

All but two window menu options, Delete and Additional Options, are available directly from DECwindows. These functions are not available on VAXuisx.

1.9.4 Resizing Windows

UIS and VAXuisx resize windows differently through the user interface. UIS lets you cross the boundary of the outline border when resizing a window. VAXuisx does not let you cross the boundary of the outline border.

For example, dragging the right border of the window past the left border produces different results in UIS and in VAXuisx. In UIS, the right border actually crosses the left border and then becomes the left border of the new, resized window. The graphics in the old window remain on the left-hand side of the left border.

In VAXuisx, DECwindows does not let you drag the right border of the window across the left border. It picks up the left side of the border and then proceeds as if you were trying to resize from the left border. The graphics in the old window appear in the new, resized window.

A Fonts

This appendix lists the three types of fonts that VAXuisx comes with.

VAXuisx comes with three types of fonts: 77 DPI (dots per inch), 100 DPI, and SIGHT fonts. VMSINSTAL places them into SYS$SYSROOT:[SYSFONT.UISX] during installation. All three fonts are VWS fonts that were converted to DECwindows format.

A.1 77 DPI Fonts

DETEKPATAAAAAAF000000000DA.DECW$FONT
DEUISPATAAAAAAF000000000DA.DECW$FONT
DTABER0003WK00GG0001UZZZZ02A000.DECW$FONT
DTABER0003WK00PG0001UZZZZ02A000.DECW$FONT
DTABER0G03CK00GG0001UZZZZ02A000.DECW$FONT

DTABER0I03WK00GG0001UZZZZ02A000.DECW$FONT
DTABER0I03WK00PG0001UZZZZ02A000.DECW$FONT
DTABER0M03CK00GG0001UZZZZ02A000.DECW$FONT
DTABER0M06OK00GG0001UZZZZ02A000.DECW$FONT
DTABER0R03WK00GG0001UZZZZ02A000.DECW$FONT

DTABER0R03WK00PG0001UZZZZ02A000.DECW$FONT
DTABER0R07SK00GG0001UZZZZ02A000.DECW$FONT
DTABER0R07SK00PG0001UZZZZ02A000.DECW$FONT
DTERMING03CK00PG0001UZZZZ02A000.DECW$FONT
DTERMINM03CK00PG0001UZZZZ02A000.DECW$FONT

DTERMINM06OK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0A00KK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0G03CK00GG0001QZZZZ02A000.DECW$FONT
DVWSVT0G03CK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0G03CK00PG0001QZZZZ02A000.DECW$FONT

DVWSVT0G03CK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0G05AK00GG0001QZZZZ02A000.DECW$FONT
DVWSVT0G05AK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0G05AK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0G05AK00PG0001UZZZZ02A000.DECW$FONT

DVWSVT0I03WK00GG0001QZZZZ02A000.DECW$FONT
DVWSVT0I03WK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0J05AK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0J05AK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0K05AK00GG0001QZZZZ02A000.DECW$FONT

DVWSVT0K05AK00GG0001UZZZZ02A000.DECW$FONT

DVWSVT0K05AK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0K05AK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0N03CK00GG0001QZZZZ02A000.DECW$FONT
DVWSVT0N03CK00GG0001UZZZZ02A000.DECW$FONT

DVWSVT0N03CK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0N03CK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0N05AK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0N05AK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0N06OK00GG0001QZZZZ02A000.DECW$FONT

DVWSVT0N06OK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0N06OK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0N06OK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0N0AKK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0N0AKK00PG0001UZZZZ02A000.DECW$FONT

DVWSVT0R03WK00GG0001QZZZZ02A000.DECW$FONT
DVWSVT0R03WK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0R07SK00GG0001QZZZZ02A000.DECW$FONT
DVWSVT0R07SK00PG0001QZZZZ02A000.DECW$FONT
DVWSVT0V05AK00GG0001UZZZZ02A000.DECW$FONT

DVWSVT0V05AK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT0V0AKK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT0V0AKK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT1G03CK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT1G05AK00GG0001UZZZZ02A000.DECW$FONT

DVWSVT1I03WK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT1J05AK00GG0001UZZZZ02A000.DECW$FONT
DVWSVT1J05AK00PG0001UZZZZ02A000.DECW$FONT
DVWSVT1K05AK00GG0001UZZZZ02A000.DECW$FONT

A.2 100 DPI Fonts

DETEKPATAAAAAAF000000000DA.DECW$FONT
DEUISPATAAAAAAF000000000DA.DECW$FONT
DWSMENU003WK00GG0001UZZZZ02B000.DECW$FONT
DWSMENU003WK00PG0001UZZZZ02B000.DECW$FONT
DWSMENU003WK01GG0001UZZZZ02B000.DECW$FONT

RCOURIRG03WK00GG0001QZZZZ02B000.DECW$FONT
RCOURIRG03WK00GG0001UZZZZ02B000.DECW$FONT
RCOURIRG03WK00PG0001QZZZZ02B000.DECW$FONT
RCOURIRG03WK00PG0001UZZZZ02B000.DECW$FONT
RCOURIRI03WK00GG0001QZZZZ02B000.DECW$FONT

RCOURIRI03WK00GG0001UZZZZ02B000.DECW$FONT
RCOURIRI03WK00PG0001QZZZZ02B000.DECW$FONT
RCOURIRI03WK00PG0001UZZZZ02B000.DECW$FONT
RCOURIRN03WK00GG0001QZZZZ02B000.DECW$FONT
RCOURIRN03WK00GG0001UZZZZ02B000.DECW$FONT

RCOURIRN03WK00PG0001QZZZZ02B000.DECW$FONT
RCOURIRN03WK00PG0001UZZZZ02B000.DECW$FONT
RCOURIRN07SK00GG0001QZZZZ02B000.DECW$FONT
RCOURIRN07SK00GG0001UZZZZ02B000.DECW$FONT
RCOURIRN07SK00PG0001QZZZZ02B000.DECW$FONT

RCOURIRN07SK00PG0001UZZZZ02B000.DECW$FONT
RCOURIRR03WK00GG0001QZZZZ02B000.DECW$FONT
RCOURIRR03WK00GG0001UZZZZ02B000.DECW$FONT
RCOURIRR03WK00PG0001QZZZZ02B000.DECW$FONT
RCOURIRR03WK00PG0001UZZZZ02B000.DECW$FONT

RCOURIRR07SK00GG0001QZZZZ02B000.DECW$FONT
RCOURIRR07SK00GG0001UZZZZ02B000.DECW$FONT
RCOURIRR07SK00PG0001QZZZZ02B000.DECW$FONT
RCOURIRR07SK00PG0001UZZZZ02B000.DECW$FONT

A.3 SIGHT Fonts

DWYSIFPJ03CK00GG0001UZZZZ02A000.DECW$FONT
DWYSIFPJ03CK00PG0001UZZZZ02A000.DECW$FONT
DWYSIFPJ03CK02GG0001UZZZZ02A000.DECW$FONT
DWYSIFPL02SK00GG0001UZZZZ02A000.DECW$FONT
DWYSINS001OK00GG0001UZZZZ02A000.DECW$FONT

DWYSINS001OK00PG0001UZZZZ02A000.DECW$FONT
DWYSINS001OK01GG0001UZZZZ02A000.DECW$FONT
DWYSINS0028K00GG0001UZZZZ02A000.DECW$FONT
DWYSINS0028K00PG0001UZZZZ02A000.DECW$FONT
DWYSINS0028K01GG0001UZZZZ02A000.DECW$FONT

DWYSINS002SK00GG0001UZZZZ02A000.DECW$FONT
DWYSINS002SK00PG0001UZZZZ02A000.DECW$FONT
DWYSINS002SK01GG0001UZZZZ02A000.DECW$FONT
DWYSINS003CK00GG0001UZZZZ02A000.DECW$FONT
DWYSINS003CK00PG0001UZZZZ02A000.DECW$FONT

DWYSINS003CK01GG0001UZZZZ02A000.DECW$FONT
DWYSINS003WK00GG0001UZZZZ02A000.DECW$FONT
DWYSINS003WK00PG0001UZZZZ02A000.DECW$FONT
DWYSINS003WK01GG0001UZZZZ02A000.DECW$FONT
DWYSINS0050K00GG0001UZZZZ02A000.DECW$FONT

DWYSINS0050K00PG0001UZZZZ02A000.DECW$FONT
DWYSINS0050K01GG0001UZZZZ02A000.DECW$FONT
DWYSINS006OK00GG0001UZZZZ02A000.DECW$FONT
DWYSINS006OK00PG0001UZZZZ02A000.DECW$FONT
DWYSINS006OK01GG0001UZZZZ02A000.DECW$FONT

DWYSINS00A0K00GG0001UZZZZ02A000.DECW$FONT
DWYSINS00A0K00PG0001UZZZZ02A000.DECW$FONT
DWYSINS00A0K01GG0001UZZZZ02A000.DECW$FONT
DWYSISS001OK00GG0001UZZZZ02A000.DECW$FONT
DWYSISS001OK00PG0001UZZZZ02A000.DECW$FONT

DWYSISS001OK02GG0001UZZZZ02A000.DECW$FONT
DWYSISS0028K00GG0001UZZZZ02A000.DECW$FONT
DWYSISS0028K00PG0001UZZZZ02A000.DECW$FONT
DWYSISS0028K02GG0001UZZZZ02A000.DECW$FONT
DWYSISS002SK00GG0001UZZZZ02A000.DECW$FONT

DWYSISS002SK00PG0001UZZZZ02A000.DECW$FONT
DWYSISS002SK02GG0001UZZZZ02A000.DECW$FONT
DWYSISS003CK00GG0001UZZZZ02A000.DECW$FONT
DWYSISS003CK00PG0001UZZZZ02A000.DECW$FONT
DWYSISS003CK02GG0001UZZZZ02A000.DECW$FONT

DWYSISS003WK00GG0001UZZZZ02A000.DECW$FONT
DWYSISS003WK00PG0001UZZZZ02A000.DECW$FONT

DWYSISS003WK02GG0001UZZZZ02A000.DECW$FONT
DWYSISS0050K00GG0001UZZZZ02A000.DECW$FONT
DWYSISS0050K00PG0001UZZZZ02A000.DECW$FONT

DWYSISS0050K02GG0001UZZZZ02A000.DECW$FONT
DWYSISS006OK00GG0001UZZZZ02A000.DECW$FONT
DWYSISS006OK00PG0001UZZZZ02A000.DECW$FONT
DWYSISS006OK02GG0001UZZZZ02A000.DECW$FONT
DWYSISS00A0K00GG0001UZZZZ02A000.DECW$FONT
DWYSISS00A0K00PG0001UZZZZ02A000.DECW$FONT
DWYSISS00A0K02GG0001UZZZZ02A000.DECW$FONT

B VAXuisx Logical Names

This appendix list the VAXuisx logical names.

VAXuisx can be customized on a per-process and system wide basis by the use of logical names. The command file SYS$STARTUP:UISX$STARTUP.COM is executed by SYSMAN during VMS system startup. This command file establishes the defaults by creating the logical name table, UISX$LOGICAL_TABLE, with the default logical name settings.

Users can copy and execute this command procedure to create a process specific logical name table by supplying the keyword "USER" as the only argument to this command procedure. In addition, VAXuisx establishes a logical name search list which will allow logical names in the users default JOB, PROCESS, GROUP and SYSTEM tables to override the settings in the default logical name table.

3.1 VAXuisx Logical Name Table

Table B–1 describes the currently defined logical names.

Table B–1

VAXuisx Logical	Default	Explanation
UISX$FLUSH	False	Flush Xlib buffer after every graphic call.
UISX$AUTO_FLUSH	True	Flush Xlib and PIXMAP buffer via timer.
UISX$AUTO_FLUSH_TIMER	"0 0:0:0.05"	VMS Delta-Time timer interval.
UISX$BATCH_COUNT	100000	Defines the maximum number of output operations that can be performed to the PIXMAP before copying to the window. The copy can also be done as a result of input or by the FLUSH TIMER.
UISX$WRITE_IMMEDIATE	False	Draw all operations to window with no PIXMAP buffering optimization.
UISX$SYNCHRONIZE	False	Cause all X11 operations to be synchronous.
UISX$BACKING_STORE	True	Creates a X11 PIXMAP which is used to guarantee the window contents and buffer drawing operations.
UISX$BACKING_STORE_CAN_FAIL	False	Allows the failure to create a backing store PIXMAP as a non-fatal error.
UISX$MATCH_SIZE	True	Causes any DPI from 74 to 79 to be forced to 77 DPI, the DPI used by VWS for most monitors (DECwindows reports as only 75 or 100 DPI).

Table B–1 (Cont.)

VAXuisx Logical	Default	Explanation
UISX$DPI_X	0	Allows the user to override the DPI setting VAXuisx uses when converting between physical units (typically centimeters) and pixel units. If set to a non-zero value this overrides the computed of the server's screen.
UIXS$DPI_Y	0	Allows the user to override the DPI setting VAXuisx uses when converting between physical units (typically centimeters) and pixel units. If set to a non-zero value this overrides the computed dots_per-inch of the servers screen.
UISX$CUSTOMIZE_COLORS	TRUE	If defined to false the user's color customization file DECW$USER_DEFAULTS: DECW$SM_ COLOR.DAT will not be read. Not reading the file results in the default color settings of white window background with black foreground.
UISX$NO_NA_SIGNAL	False	If defined to true UIS functions which are not available for VAXuisx will NOT signal an error.
UISX$NO_NYI_SIGNAL	False	If defined to true UIS functions which are not yet implemented will NOT signal an error.
UISX$ICON_NUMBER	1	Selects an icon from the icon fonts. The default icon font is built into VAXuisx.
UISX$ICON_32	UNDEFINED	Can be used to select a font for use as the 32 x 32 pixel icon image. The font must be a fixed pitch and 32 x 32 and in VWS format on the CLIENT side.
UISX$ICON_16	UNDEFINED	Can be used to select a font for use as the 16 x 16 pixel icon image. The font must be a fixed pitch and 16 x 16 and in VWS format on the CLIENT side.
UISX$DIRECT_MAP	UNDEFINED	Can be used on any system that supports a direct color X11 Visual Class of DirectColor. Forces Visual Class to DirectColor.
UISX$MONOCHROME	FALSE	Makes most applications believe they are being run on a monochrome workstation.
UISX$REQUIRE_DEFAULT	FALSE	Prevents the Workstation Data Block flagged as the default (SYS$WORKSTATION) from being closed.
UISX$REQUIRE_BANNER	TRUE	Causes the no banner and no border attributes on the window to be ignored. This prevents the problem of windows which are not under control of the DECwindows Window Manager, and thus, cannot be moved, pushed, popped, or manipulated in any manner from the human interface.

Table B–1 (Cont.)

VAXuisx Logical	Default	Explanation
UISX$USE_KEYSYM	FALSE	If defined to true, the keycodes passed back to the user from a uis$set_kb_ast will be in the form of the default DECwindows keysym's. This also allows the user to use the currently mapped DECwindows keyboard to return keycodes. **Note: Compose state cannot be entered when this logical is true in any UISX windows.**

Note:

Using UISX$SYNCHRONIZE dramatically reduces the performance of VAXuisx.

When both UISX$FLUSH and UISX$SYNCHRONIZE are FALSE, your application may appear to have not finished outputting to the screen. The reason may be that the DECwindows/X11 Client has not yet sent some buffered commands to DECwindows/X11 SERVER.

Mouse movement, button events, or window movement may cause VAXuisx to flush the queue for the application. By setting either UISX$FLUSH or UISX$SYNCHRONIZE to TRUE, you will guarantee that VAXuisx sends all operations to DECwindows/X11.

Native UIS drawing operations appear smooth. DECwindows/X11 drawing operations may seem "pulsing" or "bursty." You can make DECwindows/X11 drawing operations more smooth by flushing output after each call by setting UISX$FLUSH to TRUE.

Glossary

ATB: See Attribute Block.

Attribute Block (ATB): A UIS data structure that describes the appearance of any graphic object an application program creates.

Backing Store: A term describing the way UIS preserves the contents of a window when parts or all of it is obscured. You can have UIS do backing store in a number of ways. The most common is by having UIS make a bit map backup of the window or the hardware drawing commands to redraw the window. (Bit map backup is the only mechanism that provides an upper bounds on the amount of resources required to provide window integrity.)

Bit Map: A table describing each item in a related set if items.

Bitmap: A sequence of bytes representing a printing character.

Cap Butt: A cap style whose line is square at the endpoint (perpendicular to the slope of the line) with no projection beyond.

Cap Not Last: A cap style whose line is square at the endpoint (perpendicular to the slope of the line) with no projection beyond. The endpoint is not drawn for a line width of zero.

Cap Projecting: A cap style whose line is square at the end, but whose path continues beyond the endpoint to a distance that equals half the line width. Cap Projecting is equivalent to a line width of zero.

Cap Round: A type of cap style whose line has a circular arc. The diameter of the arc is equal to the width of the line, centered on the endpoint. Cap Round is equivalent to Cap Butt for a line width of zero.

Cap Style: Defines how the endpoints of a path are drawn. Four cap styles are available in DECwindows/X11: Cap Not Last, Cap Butt, Cap Round, and Cap Projecting. VAXuisx uses *only* Cap Butt.

Client: An X11 application. A client sends the commands to an X11 Server, which displays them.

DECwindows: The term for Digital's implementation of the X Window System™, Xlib, XToolkit, Window Manager, Session Manager, User Interface Language, and Compound Document Architecture. People within Digital use DECwindows interchangeably for X11 and the X Window System.

DECwindows Toolkit: High-level library routines that allow applications to create and manage a user interface. Using the DECwindows Toolkit, routines can create, modify, and control interface objects such as menus, scroll bars, and buttons. The DECwindows Toolkit routines call Xlib routines to perform fundamental input and output functions. The DECwindows Toolkit includes Digital's implementation of the toolkit for X Window System.

Digitizer: Name for the tablet that maps its input directly into a single window on UIS. Users normally use a digitizer when working with CAD/CAM-type applications.

Direct-Color System: A system that decomposes the pixel color value into three separate subfields for indexing. The first subfield indexes the RED colormap. The second subfield indexes the BLUE colormap. And the third subfield indexes the GREEN colormap. The user can change the values in the three colormaps dynamically.

DOP: See Drawing Operation Packet.

Drawing Operation Packet (DOP): The low-level interface to GPX-based systems under UIS.

Event Queue: A queue of events the application handles. Each application has a unique event queue.

Flush: To cause the processing of items within a buffer.

In VAXuisx, to force the transmission of graphic commands from the Xlib buffer to the DECwindows/X11 Server for execution.

GC: See Graphics Context.

Generic Encoding: A device-independent code UIS uses for its calls. The UIS display list uses generic encoding internally to store the UIS commands. UIS also uses generic encoding as the on-disk structure for its output.

Glyph: The pictorial representation of a font character represented internally by a bitmap

Grapics Context (GC): A data structure in X11 that contains the attributes DECwindows uses when drawing graphics primitives. An unlimited number of GCs can be available for each physical display (depending on available resources).

Graphic Text: Text the user can scale, rotate, and/or shear (slant). VAXuisx treats graphic text as a graphic object because it requires special operations to cause the changes requested.

Hardcopy UIS (HCUIS): Provides both a program interface and the user interface to RENDER for reading and writing UIS generic encoding. HCUIS also provides conversions from generic encoding to various output formats such as HPGL, SIXEL, ReGIS, and PostScript.

HCUIS: See Hardcopy UIS.

Join Bevel: A type of join style in which the corner has cap butt end point styles with the triangular notch filled.

Join Miter: A type of join style in which the outer edges of two lines extend to meet at an angle. (If the angle is less than 11 degrees, however, a Join Bevel join style is used.) VAXuisx uses *only* Join Miter.

Join Round: A type of join style in which the corner is a circular arc whose diameter equals the line width. Join Round is centered on the joint point.

Join Style: A term defining how corners are drawn for wide lines. Three types of join styles are available: Join Miter, Join Round, and Join Bevel. VAXuisx uses *only* the Join Miter join style.

Miter: A joint made by beveling each of two surfaces at an angle and fitting them to form a corner.

NTSC Color Convention: The television standard conversion of RGB into chroma.

Open Software Foundation (OSF): A nonprofit organization several companies created to provide a standard environment for applications. Digital is a founding member of OSF.

OSF: See Open Software Foundation.

PIXMAP: A collection of bit maps, one per plane, that are available on the hardware display.

Pseudo-Color System: A system that indexes the pixel color value to the hardware colormap.

QAR: See Quality Assurance Report.

Quality Assurance Report (QAR): A form for people at field test sites to use for reporting bugs they encounter during the field test.

Server: Accepts commands from Clients and displays output on a workstation, PC, or other output device. The X11 Server also sends input from the keyboard or mouse to the Client.

Software Problem Report (SPR): A form customers use for reporting bugs to Digital.

SPR: See Software Problem Report

True-Color System: A system with Read-only colormaps.

UIS: See User Interface Services.

User Interface Services (UIS): The runtime library interface for VWS. UIS is also used interchangeably with VMS Workstation Software (VWS).

VMS Workstation Software (VWS): A window and graphic system for the VMS operating system. VWS is also used interchangeably with User Interface Services (UIS), which was the original name for VWS.

VWS: See VMS Workstation Software.

Writing Mode: The attribute that assigns the mode of writing graphics or text. Writing Mode determines how a text or graphics routine uses the writing and background colors index to display a graphic object.

X Window System (X11): The name for a window system MIT and Project Athena developed. It is the official name for X11.

X11: The generic term used for the X Window System.

Xlib: The standardized, procedural interface to X11.

XToolkit: A collection of routines that implements an object-oriented user interface for X11.

Index

P

S

T

U

V

W