VAXuisx Release Notes

Order Number: AA-PC3EA-TE

Software Version:	VAXuisx V1.0
Operating System:	VMS V5.3 or above
	You must have DECwindows installed in order for VAXuisx to run.

August 1990

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

Preface

This manual provides supplemental information about V1.0 of the VAXuisx Runtime Library for VMS (VAXuisx). If you have not already done so, please read the *Read Me First* information included with your documentation.

This manual is for graphics programmers and users who should know about VAXuisx performance and restrictions. All users should read this document before using the VAXuisx Runtime Library for VMS.

The manual is divided into two chapters: one on restrictions and another on performance issues.

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 your system.
- *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 converting UIS to DDIF.
- *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 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 B, 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 Changes and Restrictions

VAXuisx has some restrictions other than those imposed by UIS which may apply to your application. This chapter describes those restrictions.

1.1 UIS Features Not Implemented in VAXuisx

VAXuisx does not implement all the UIS features. For example, UIS provides a tablet/digitizer that can map to a window. UIS confines the cursor to a particular window and the application controls the aspect ratio and the resolution of the input. X11 has no mechanism for mapping tablet input to a window. VAXuisx does not emulate the UIS tablet/digitizer, but always treats the tablet as if it were a mouse.

Some UIS features, such as shared color maps, may be implemented in later versions of VAXuisx. Other features will never be implemented in VAXuisx. Those features that are not implemented for this version of VAXuisx include:

- DOPs
- The VWS QIO interface
- Digitizer/tablet support
- Sound key click
- Shared colormaps
- The VWS Display Manager
- Specific VWS utilities such as Color Print Screen
- The VWS terminal emulators
- Kernel Mode access and process permanent UIS structures
- The VWS look and feel

1.2 UISX$STARTUP Should Not Be Run on VWS Systems

VAXuisx uses a startup file named UISX$STARTUP.COM. This command procedure should not be run on a system running native VWS. If it is, some very strange behavior may result.

This file is run at system startup time. When the file runs, it checks the SYSGEN parameter WINDOW_SYSTEM. If this parameter is set to 0 (no windowing system) or 1 (DECwindows), the command file will continue. If this parameter is set to 2 (VWS), it will exit.

If this check is ever modified and the command procedure run on a VWS workstation, VWS applications will not run. The reason for this is the fact that VAXuisx defines UISSHR (the UIS shared image) to be UISXSHR (the VAXuisx shared image). This definition is what enables VWS applications to be run without recompiling and relinking.

Note: **Using the SET DISPLAY method to create a workstation device in order to point SYS$WORKSTATION towards may lead to the following VAXuisx error message if DECNET is shut down after VAXuisx has run its startup file.**

```
"UISX-E-OPEN_ERROR, error opening display"
```

This is due to the fact that, by default when a SET DISPLAY is done to create a workstation device the default transport used is DECNET, and unless DECNET has not been started or your system startup files have the DECW$IGNORE_DECNET logical defined to be TRUE, then SET DISPLAY uses the default DECNET transport in the UISX$STARTUP.COM procedure.

In order to check which workstation device VAXuisx is using and which transport is being used and to ensure that you use the LOCAL transport when DECNET is shut off follow these steps:

```
$ SHOW LOGICAL/TABLE=UISX$LOGICAL_TABLE SYS$WORKSTATION

   "SYS$WORKSTATION" = "_WSA2:"

$ SHOW DISPLAY _WSA2:

    DEVICE:         WSA2:    [exec]
    NODE:           0
    TRANSPORT:      DECNET
    SERVER:         0
    SCREEN:         0

$ SET DISPLAY _WSA2:/TRANSPORT=LOCAL
```

This will reset your transport to use the LOCAL transport and will allow your application to continue to work under VAXuisx.

1.3 SYS$WORKSTATION

Depending on where your application is being run from, you have a few options as to how to define this logical. By default on systems with the SYSGEN parameter WINDOW_SYSTEM set to either 0 or 2, VAXuisx will use the logical DECW$DISPLAY in order to point to the workstation device. On systems currently running DECwindows, VAXuisx will actually create a display on which to run the application.

There are three ways in which you can run your application under VAXuisx to a DECwindows server:

- Define the SYS$WORKSTATION logical to point at the logical DECW$DISPLAY. Note that the DECW$DISPLAY logical is defined only in the JOB logical name table, which means applications run in detached mode will parse the DECW$DISPLAY logical.

- Define the SYS$WORKSTATION logical to point at display which you can create by using the DCL command SET DISPLAY /CREATE. This will create a DECwindows WSA device which you can define your SYS$WORKSTATION logical in the UISX$LOGICAL_TABLE to point to.
- Define the SYS$WORKSTATION logical to point at the string node::server.screen.

  Node = DECNET node in which the graphics output can be displayed
  Server = Server number for that node
  Screen = Screen for the server

There are examples of each possible way to define your SYS$WORKSTATION logical in the UISX$STARTUP.COM file. If you do decide to redefine the SYS$WORKSTATION logical in any other form than the default make sure to use the /TABLE=UISX$LOGICAL_TABLE in your definition command as follows:

```
DEFINE /SYSTEM /TABLE=UISX$LOGICAL_TABLE SYS$WORKSTATION ...

  or

DEFINE /PROCESS /TABLE=UISX$LOGICAL_TABLE SYS$WORKSTATION ...
```

1.4 Linking UIS Applications with UISXSHR

If you link a UIS application program on a system running VAXuisx, the program may have problems running on other systems. This is because VAXuisx redefines the UIS shareable library (UISSHR) as SYS$SHARE:UISXSHR.EXE, which requires the DECwindows shareable for image activation.

When a UIS program is linked, it links in symbols from UISSHR. UISXSHR.EXE includes symbols from the DECwindows shareable library (DECW$DWTLIBSHR.EXE). When the program is run, it will attempt to resolve all symbols, including those defined in DECW$DWTLIBSHR.EXE. If the image file is copied to a system without DECwindows, or with a different version of the DECW$DWTLIBSHR shareable library, the program will not run because it cannot be resolved correctly.

To avoid this problem, LINK your UIS application programs against the stub UIS shareable image (SYS$SHARE:UISSHR.EXE):

```
$!
$ LINK program, SYS$INPUT /OPTIONS
 SYS$SHARE:UISSHR /SHARE
$!
```

1.5 VMS V5.1 and V5.2 are Not Supported

Due to problems with VMS DECwindows Version 1 (VMS V5.1 and V5.2) VAXuisx is supported only by VMS Version 5.3 and later. When run using earlier versions of VMS DECwindows as either client or server, programs may randomly exit without reason, or with XLIB FATAL I/O errors. This usually happens in applications which execute at AST level for significant amounts of time while doing interactive operations using the mouse.

1.6 Fonts after Installation

You may need to restart your DECwindows server after installation of VAXuisx to pick up the new fonts. The server may be restarted without rebooting your system by entering:

```
$@SYS$MANAGER:DECW$STARTUP RESTART
```

from a suitably privileged account. The fonts should be reloaded by logging out from the session and logging back in, but it has been found that this is not always the case.

1.7 Running VAXuisx Applictions Detached

If a VAXuisx application is run as a detached process with an error log file specified, (ie. RUN/DETACHED/ERROR=ERROR_FILE.LOG FOO.EXE), a log file will be created if the logical UISX$CUSTOMIZE_COLORS is set to TRUE, it's default value, whether or not an error occurs. This behavior is due to a problem in the DECwindows Resource Manager. VAXuisx uses the Resource Manager to obtain the default window's colors chosen by the user.

There are several possible ways to work around this problem. If the logical UISX$CUSTOMIZE_COLORS is set to FALSE, VAXuisx will not call the DECwindows Resource Manager and the error file will not be created. Note that this logical must be defined in EXECUTIVE mode in either the group or the system logical name table for the detached process to pick it up.

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE UISX$CUSTOMIZE_COLORS FALSE
   or
$ DEFINE/GROUP/EXECUTIVE_MODE UISX$CUSTOMIZE_COLORS FALSE
```

The draw back from using this method is that it will disable VAXuisx from inheriting the user selected DECwindows window colors.

If you are concerned with the amount of disk space which will be consumed, with this error log file, you may limit this by setting a version limit on the error file.

```
$ SET FILE/VERSION_LIMIT=3 ERROR_FILE.LOG
```

When the process is run and the error file is currently at its version limit, (three copies already exist), the older version, (the one with the lowest version number), will be purged and a new file will be created.

If you are using this method, you should be aware of any log files which contain relevant data that you need to retain. To prevent lose of these files you must be very careful to rename the needed files to file names which will not be automatically purged.

If you do not want this error log file to ever be created, whether an error occurs or not, you can create an empty error log file with a version number of 32767.

```
$ CREATE ERROR_FILE.LOG;32767
CTRL-Z $
```

This is the highest version number allowed by VMS, thus prohibiting the create of a new file.

1.8 Tablet Support

The tablet works only as a mouse because the X Window System™ does not support the tablet in digitizing mode. None of the tablet functions work under VAXuisx. Calls to UIS$GET_TB_INFO will always return **FALSE**, indicating that no tablet is available.

1.9 UISX$SYNCHRONIZE

Using UISX$SYNCHRONIZE for interactive applications that use ASTs intensively is not recommended and can cause server deadlocks, Xlib I/O errors and server hangs.

1.10 Shared Color Maps and Attributes

Shared Color Maps are not implemented in Version 1.0. All colormap attributes are ignored.

1.11 Keyboard Restrictions

Because VAXuisx is layered on top of DECwindows and is running under the control of the DECwindows Session and Window Managers, the F1 through F5 functions keys do not have their VWS definitions. These keys perform the functions assigned to them by DECwindows.

The fallback mouse keys do not work using the VWS mechanism described in the following table. Instead DECwindows provides mouse emulation by pressing CTRL F3 at which point the WAIT light will be turned on and the keys in the table will operate as if CTRL/Shift were also depressed.

Key Sequence	Key Code
CTRL/Shift Select	Select mouse button
CTRL/Shift Prev Screen	Middle mouse button
CTRL/Shift Next Screen	Right mouse button
CTRL/Shift ↑	Move mouse up one pixel
CTRL/Shift ↓	Move mouse down one pixel
CTRL/Shift ←	Move mouse left one pixel
CTRL/Shift →	Move mouse right one pixel

1.12 UIS Window Options Menu

Under VAXuisx there will be no MENU icon in the upper left hand corner of each window for a window options menu. This is due to the inability of VMS DECwindows to allow applications to customize windows in this manner. For VAXuisx, most of the options are replaced by the DECwindows Window Manager. Following is a list of options available through the VWS MENU icon in the window options menu, and its DECwindows replacement, if one exists:

```
Push Behind         -> Use the icon in the VAXuisx banner for push
Pop in Front        -> Press button in banner of window to pop
Delete              -> There is no equivalent
Change the Size     -> Use the resize icon in the VAXuisx banner
Shrink to Icon      -> Use the shrink icon in the VAXuisx banner
Additional Options  -> There is no equivalent
Exit from this Menu -> No need for equivalent
```

1.13 Window Placement

The full window placement logic from VWS has not 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*, and *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 the first window created in the region 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 as best as possible between the edges. If no placement is specified, top and centered 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. Rather, if an area specified currently has a viewport visible, an attempt will be made to place the window in a southeasterly fashion as the DECwindows default placement does. If the window will fit on the screen at that point, it is placed there. If it cannot be, an attempt is made to move the window either west or north dependin on which portion of the viewport would have been offscreen by using the southeasterly approach.

1.14 Borderless and Bannerless Windows

By default, all windows created by VAXuisx will have borders and banners, regardless of the attributes requested in the UIS$CREATE_WINDOW routine. This behavior is controlled by the logical UISX$REQUIRE_BANNER.

If this logical is defined to be **FALSE**, which is not the default, VAXuisx will create DECwindows windows which do not have a banner. In this case, the window is not under the control of the DECwindows Window Manager. Any windows not under control of the window manager are incapable of being manipulated from the user interface. That is, they may not be moved, pushed, popped, resized, etc. by the user utilizing the mouse or tablet.

1.15 UIS$PRESENT - New, Optional Parameter

VAXuisx has added a new, optional parameter at the end of the parameter list for UIS$PRESENT. This parameter returns TRUE (1) if the program is running from VAXuisx. It returns FALSE (0) if the program is running from native UIS. For more information on this routine and its new format, refer to the *VWS Release Notes*.

1.16 UIS$CREATE_TERMINAL

You cannot create a TK-type terminal under VAXuisx. Trying to create one through the use of the routine UIS$CREATE_TERMINAL causes a fatal error in your program. VAXuisx also displays the following error message:

```
%UISX-F-NOTK, Cannot create a TK terminal device
```

You must specify WT as the TERMTYPE argument. However, instead of creating a WT-type terminal, UIS$CREATE_TERMINAL creates a DECterm terminal. UIS$CREATE_TERMINAL currently ignores the terminal and placement attributes (DECwindows terminal emulator).

When VAXuisx is run on a non-workstation, the psuedo-terminal drivers must be loaded and the terminal controller process must be started on the client system. You may need to use the CREATE/TERMINAL/CONTROLLER DCL command.

1.17 Filled and Unfilled Circle Edges

Filled and unfilled circles may not overlap exactly. Some of the edge pixels are considered inside the circle and some are considered outside. This is usually not a problem unless the edge of a circle is drawn by an application prior to drawing the filled portion of the same circle. This difference is the result of the way X11 implements circles.

1.18 Resize from AST

VAXuisx does not currently handle UIS$RESIZE_WINDOW correctly when called with null parameters from a resize AST. The world coordinate system is not scaled to the new window but remains unchanged.

1.19 UIS$GET_WS_COLOR and UIS$GET_WS_INTENSITY

If UIS$GET_WS_COLOR or UIS$GET_WS_INTENSITY are not explicitly passed a WD_ID (that is, asking for a realized color), VAXuisx uses SYS$WORKSTATION.

1.20 Rendering UIS Files

To render UIS files produced by running your application with VAXuisx, you should install HCUIS and the VWS fonts supplied with the VWS 4.3 installation kit. For installation instructions, refer to the *Installation Guide for VWS Software and Migration Tools*.

1.21 Linking C Programs

You cannot link C programs with the logical LNK$LIBRARY defined as VAXCRTL when using VAXuisx. Attempting to do this will result in repeated occurrences of Multiply Defined Symbol errors. To alleviate this problem, deassign the logical LNK$LIBRARY.

2 Performance

This chapter discusses some of the performance differences that exist between UIS and VAXuisx.

2.1 Backing Store Performance

VWS guarantees the integrity of all windows by maintaining a bitmap backup (backing store) of each window. Therefore, VWS applications do not need to be concerned with maintaining the contents of their windows. This is almost as great as having no backing store. This feature can be defeated (DEFINE UISX$WRITE_IMMEDIATE TRUE), and all operations are done immediately to both the PIXMAP and the window. This option will cause performance to be at least 50% less for most output. The number of operations that are written to the PIXMAP for each copy from PIXMAP to window can be set (DEFINE UISX$BATCH_COUNT **integer_value**) and can be used to limit the number of batched operations. Large values allow better performance, small values (including 0 or 1) allow lower performance but smoother drawing operations.

Backing store can be completely disabled, (DEFINE UISX$BACKING_STORE FALSE), which will cause operations to be written only to the window and no off screen PIXMAP can be allocated. This provides the best possible performance, but the contents of windows can be lost by occlusion or when the window is in the icon state.

2.2 Using FLUSH and AUTO FLUSH

By default, a timer event is generated which causes any operations not copied to the screen from the PIXMAP and any operations in the Xlib drawing buffer to be flushed. This feature is called Auto-Flush. It can be disabled (DEFINE UISX$AUTO_FLUSH FALSE) which will disable the timer. The timer interval can be set (DEFINE UISX$AUTO_FLUSH_TIMER "**delta_time**") to a VMS delta-time value. Small time values provide smoother output, large time values provide higher throughput.

In addition to Auto-Flush, the X11 output buffers can be flushed after each output operation (DEFINE UISX$FLUSH TRUE). This will cause a performance decrease of a variable amount and is generally not needed but can be useful for debugging.

2.3 Using UISX$SYNCHRONIZE

UISX$SYNCHRONIZE dramatically affects the performance of VAXuisx. This feature is useful only for limited reasons, mostly for debugging.

For further information on UISX$FLUSH and UISX$SYNCHRONIZE, refer to Appendix B, "Logical Names", in the *VAXuisx User's Guide*.

2.4 Drawing Wide Lines

When you use VAXuisx, there will be a significant performance difference between thin lines (lines that are one pixel wide) and wide lines (lines that are wider than one pixel). The difference occurs because VAXuisx special cases thin lines and allows X11 to draw them using whatever line algorithm is implemented for the graphics hardware. Wide lines must use the line algorithm specified by X11 and thus cannot take advantage of the graphics hardware accelerators.

2.5 Making Frequent Changes to Attribute Blocks

If your application frequently changes values within an attribute block, you will experience a significant degradation in drawing performance. For example, the degradation will occur if your application changes an attribute, draws a single object, an changes attribute, draws a single object, and so on. This is an X11 behavior with changing graphic attributes.

2.6 Using Patterned Fill Styles

Use of solid fill styles (foreground or background) will yield a better performance than with the use of other fill styles. This is due to the method that must be used to implement these fill styles in VAXuisx. For solid patterns, X11 provides an optimized code path which is used by VAXuisx. For non-solid patterns, VAXuisx must build an X11 PIXMAP and use it in the fill operation.

2.7 Graphics Text

Graphics text is any text which is sloped, slanted, rotated, scaled, or has spacing other than the default. X11 does not allow any of this manipulation of characters. Therefore, implementation of graphics text in VAXuisx requires creating an X11 image which contains the string and copying this string to the VAXuisx window and, if enabled, the backing store PIXMAP.

The image is compiled by performing the appropriate geometric operation on each bit that forms the letter, for each character in the string and writing the result to the image. This operation is very expensive. As a result, performance of graphics text is slower than that of normal, non-graphics text.

The performance rate of graphics text depends on which type of character manipulation you are using. Below is a list of the types of graphics text listed from the best performance to the worst performance. Remember that combining any of these characteristics will perform, at best, at the rate of the slower of the types. It is likely that performance will be somewhere below the worst performer.

Graphics Text Type (listed best to worst performer)

- Non-default spacing
- Slanted text
- Rotated text
- Scaled text
- Sloped text

2.8 Writing Modes

Many of the UIS writing modes are not directly available in X11. In order to get the expected result for these writing modes VAXuisx must use a combination of writing mode, foreground color, background color, plane mask, and fill pattern. It is also at time necessary to draw the object multiple times to achieve the proper output. Because of this, some UIS writing modes are slower than others when using VAXuisx.

Performance of writing modes also depends on whether the output primitive is text or graphics. Refer to Table 2–1 which lists the various writing modes and indicates which ones result in a significant performance degradation for both text and graphics output.

Table 2–1

Writing Mode	Text	Graphics
Erase		
Erase Negate		
Overlay		
Overlay Negate	X	X
Replace		
Replace Negate		
Complement		
Transparent		
Bit Clear		X
Bit Clear Negate	X	X
Bit Set		X
Bit Set Negate	X	X
Copy	X	
Copy Negate	X	

2.9 Image Performance

Image performance in VAXuisx is significantly slower than in native UIS. This performance degradation is due to the conversion that must take place in order to move image data from the UIS image format to X11 image format. To do this, each pixel must be moved individually with any pixel replication and color translation occurring during the move. The pixel replication and color translation do not require significant amounts of overhead, but going through the image one pixel at a time is a time consuming operation.

A Unsupported Routines

VAXuisx does not support some routines found in the VWS User Interface Services (UIS). This appendix lists the routines VAXuisx does not support, the severity of the error VAXuisx returns, and an explanation of the error.

Routine	Severity	Explanation
UIS$CREATE_TB	Error	Create tablet data block
UIS$DELETE_TB	Error	Delete tablet data block
UIS$DISABLE_TB	Warning	Disable as the active digitizer
UIS$ENABLE_TB	Warning	Enable as the active digitizer
UIS$GET_TB_POSITION	Error	Get tablet position
UIS$SET_ADDOPT_AST	Informational	Specify additional options
UIS$SETUP	Error	Invokes the Workstation Options Menu.
UIS$SOUND_CLICK	Informational	Sound key click
UISDC$ALLOCATE_DOP	Fatal Error	Allocate a DOP
UISDC$EXECUTE_DOP_ASYNCH	Fatal Error	Execute DOP immediately, but do not wait for completion
UISDC$EXECUTE_DOP_SYNCH	Fatal Error	Execute DOP immediately and wait for execution
UISDC$LOAD_BITMAP	Fatal Error	Load a bitmap as a font
UISDC$QUEUE_DOP	Fatal Error	Queue DOP for execution