

Copyright (c) Abstract Hardware Limited 1991, 1994

Copyright (c)

2.10.8 XSetColours	78
2.10.9 XSetDashes	

ha ter 1

Intr du li n

1.1 Th ML int rfac

We have implemented an ML interface to Xlib, the industry standard C interface for X at the lowest level, and which is widely used as the basis for many toolkits. We provide all the major function groups, so that this interface can be used to implement fully functional complex applications. We also provide a of geometric functions for handling points and and a of functions for performing logical operations on plane masks and pixel values.

Xlib is now widely documented, with many goo and programming manuals available.

We provide our version of the Xlib manual with ML signatures and types, and a more functional style to the programming interface.

We provide ML example programs to show the functionality of the ML interface to Xlib. These examples range from simple line drawing applications through to colour examples and a mini text editor showing how to with The full s.3214 0 Td (an7.7474 0 Td (of)Tj 11.9785 0 Td (the)Tj 17.805 0 Td (struc4 0 Td

Datatypes are used where possible so that arguments are strongly typed and pattern matching may be used for returned values, this is especially useful for pattern matching the different subtypes of events. Abstypes are only used for the X resource types which have no meaningful textual representation.

The functions have been made more functional. Where an Xlib function modified its arguments, this has been changed so that the function returns new, modified copies of the arguments. Where values were passed in partially filled-in structures with OR-ed bit masks, now the programmer uses constructors to make the list of v

ha ter 2

Fun ■i n R f r n

.1 Colours, Pix ls and RGB values

2.1.1 And, Or, Xor, Not, >>, <<

Types:

```
infix And Or Xor >> <<

val Not: int -> int
val And: int * int -> int
val Or: int * int -> int
val Xor: int * int -> int
val >> : int * int -> int
val << : int * int -> int
```

Description:

These functions provide useful arithmetic operations on ints representing pixel values and plane masks, which, in X, are unsigned 32-bit quantities. The least significant bits of these

quantities are on the right, and

And, Or and Xor perform bit

Not performs bitwise negation

a >> b returns a shifted b bits to the left, where

If negative values, or values greater than 32 bits, an exception Range is raised.

2.1.2 BlackPixel, WhitePixel

Types:

```
val BlackPixel: unit -> int
val WhitePixel: unit -> int
```


#RRGGBB (8 bits each)

#RRRGGGBBBB (12 bits each)

#RRRRGGGGBBBB (16 bits each)

The R, G, and B represent single hexadecimal digits (both uppercase)

Syntax:

```
val depth = DefaultDepth();
```

2.2.5 **XCreateColormap**, **XCopyColormapAndFree**, **XFreeColormap**,

Syntax:

```
XInstallColormap cmap ;  
XUninstallColormap cmap ;  
val cmaps = XListInstalledColormaps w ;
```

Arguments:

cmap Speci es the colormap.
w Speci es the window that determines the screen.

Description:

The **XInstallColormap** function installs the speci ed colormap for its associated screen. All windows associated with this colormap immediately display with true colours. You associate the windows with this colormap when you created them by calling **XCreateWindow**, **XCreateSimpleWindow**, **XChangeWindowAttributes** or **XSetWindowAttributes**. If the specified colormap is already an installed colormap, the server generates a **ColorMapNotify** event.

property	Specifies the property atom.
stdmaps	Specifies the XStandardColormaps to be used
maps	Returns the XStandardColormap

Argument Type:

```
datatype XStandardColormap = XStandardColormap of { colormap: Colormap,
                                                    redMax: int,
                                                    redMult: int,
                                                    greenMax: int,
                                                    greenMult: int,
                                                    blueMax: int,
                                                    blueMult: int,
                                                    basePixel: int,
                                                    visual: Visual }
```

Argument Description:

The colormap member is the colormap created by the **XCreateColormap** function. The redMax, greenMax, and blueMax members give the maximum red, green, and blue values, respectively. Each colour coefficient ranges from zero to its max, inclusive. For example, a common colormap allocation is 3/3/2 (3 planes for red, 3 planes for green, and 2 planes for blue). This colormap would have redMax = 7, greenMax = 7, and blueMax = 3. An alternate allocation thately

The **XGetRGBColormaps** function returns the RGB colormap definitions stored in the specified property on the named window. If the property exists, is of type **RGB_COLOR_MAP**, is of format 32, and is long enough to contain a colormap definition (if the visual is not present, **XGetRGBColormaps** assumes the default visual for the screen on which the window is located), **XGetRGBColormaps** returns the integer 0.

hotspot Specifies the x and y coordinates, which

.4 Display Specifications

2.4.1 AllPlanes

Types:

```
val AllPlanes: int
```

Syntax:

```
val planeMask = AllPlanes ;
```

Description:

AllPlanes is a value with all bits set to 1 and is suitable for use in a plane mask argument to a function.

2.4.2 BitmapBitOrder

Types:

```
val BitmapBitOrder: unit -> ImageOrder
```

Argument Type:

```
datatype ImageOrder = LSBFirst | MSBFirst
```

Syntax:

```
val order = BitmapBitOrder() ;
```

Description:

The **BitmapBitOrder** function returns **LSBFIRST** or **MSBFIRST** to indicate whether the leftmost bit in the bitmap as displayed on the screen is the least or most significant bit in the bytes comprising the data.

2.4.3 BitmapPthead

Types:

```
val BitmapPad: unit -> int
```

Syntax:

```
val pad = BitmapPad() ;
```

Description:

The **BitmapPad** function returns the number of bits that each scanline must be padded.

2.4.4 BitmapUnit

Types:

```
val BitmapUnit: unit -> int
```

Syntax:

```
val scanline = BitmapUnit();
```

Description:

The **BitmapUnit** function returns the size of a bitmap's scanline unit in bits.

2.4.5 ByteOrder

Types:

```
val ByteOrder: unit -> ImageOrder
```

Argument Type:

```
datatype ImageOrder = LSBFirst | MSBFirst
```

Syntax:

```
val order = ByteOrder;
```

Description:

The **ByteOrder** function specifies the val6ot16eByte order for images for each scanline unit in XY

2.4.7 ColormapExists,

2.4.9 DefaultVisual

T

2.4.12 DisplayPlanes

Types:

```
val DisplayPlanes: unit -> int
```

Syntax:

```
val planes = DisplayPlanes();
```

Description:

The **DisplayPlanes** function returns the depth of the root window of the screen.

2.4.13 DisplayString

Types:

```
val DisplayString: unit -> string
```

Syntax:

```
val s = DisplayString();
```

Description:

The **DisplayString** function returns the string that was passed to **XOpenDisplay** when the current display was

2.4.15 DoesSaveUnders

Types:

```
val DoesSaveUnders: uni t -> bool
```

Syntax:

```
val su = DoesSaveUnders() ;
```

2.4.18 MaxCmapsOfScreen

Types:

```
val MaxCmapsOfScreen: unit -> int
```

Syntax:

```
val n = MaxCmapsOfScreen();
```

Description:

The **MaxCmapsOfScreen** function returns the maximum number of installed colormaps supported by the screen.

2.4.24 VendorRelease

Types:

```
val VendorRelease: unit -> int
```

Syntax:

```
val n = VendorRelease();;
```

Description:

The **VendorRelease** function returns a number from the server.

This function uses these GC components: function, plane-mask, subwindow-mode, graphics-exposures, clip-origin, and clip-mask.

The **XCopyPlane** function uses a single bit plane of the specified source rectangle combined with the specified GC to modify the specified rectangle of dest. The drawables must have the

Description:

XDrawArc draws a single circular or elliptical arc, and **XDrawArcs** draws multiple circular or elliptical arcs. Each arc is specified by

endpoi

2.5.5 XDrawLine, XDrawLines, XDrawSegments

Types:

```
val XDrawLine: Drawable -> GC -> XPoint -> XPoint -> unit  
val XDrawLines: Drawable -> GC -> XPoint list -> CoordMode -> unit  
val XDrawSegments: Drawable -> GC -> (XPoint * XPoint) list -> unit
```

Syntax:

```
XDrawLine d gc point1 point2 ;  
XDrawLines d gc points mode ;  
XDrawSegments d gc segments ;
```

Arguments:

d	Specifies the drawable.
gc	Specifies the GC.
mode	Specifies the coordinate mode. You can pass CoordModeOrigin or CoordModePrevious .
points	Specifies a list of points.
segments	

2.5.6 XDrawPoint, XDrawPoints

Types:

```
val XDrawPoint: Drawable -> GC -> XPoint -> unit  
val XDrawPoints: Drawable -> GC -> XPoint list -> CoordMode -> unit
```

Syntax:

```
XDrawPoint d gc point ;  
XDrawPoints d gc points mode ;
```

Arguments:

d Specifies the dra

.n

For fonts defined with 16-bit linear indexing and used with **XDrawString**, each 8-bit character is represented by two bytes.

area	Specifies the bounding rectangle of the area. The x and y coordinates, which are relative to the origin of the drawable, specify the upper-left corner of the bounding rectangle. The width and height are the major and minor axes of the arc.
angle1	Specifies the start of the arc relative to the three-o'clock position.

The `Il-rule` of the GC controls the filling behavior of self-intersecting polygons.

This function uses these GC components: foreground, background, tile, stipple, tile-stipple-origin, function, plane-mask, Il-style, Il-rule, subwindow-mode, clip-origin, and clip-mask.

For each arc, `XFillArc` or `XFillArcs` fills the region closed by the infinitely thin path described by the specified arc and, depending on the arc-mode specified in the GC, one or two line segments. For `ArcChord`, the single line segment joining the endpoints of the arc is used. For `ArcPieSlice`, the two line segments joining the endpoints of the arc with the center point are used. `XFillArcs` fills the arcs in the same order as the list. For any given arc, `XFillArc` and `XFillArcs` do not draw a pixel more than once. If regions intersect, the intersecting pixels are drawn multiple times.

Both functions use these

components: foreground, background, tile, stipple, tile-stipple-origin, function, plane-mask, Il-style, arc-mode, subwindow-mode, clip-origin and clip-mask.

"Not a window"

Attempt to use a pixmap **Drawable** as
a window ~~useWindowable~~

"Not a pixmap"

Attempt to use

```
val ShiftDown: Modifier list -> bool
val ControlDown: Modifier list -> bool
```

Syntax:

```
ShiftDown modifiers
ControlDown modifiers
```

Arguments:

modifiers Specifies the modifiers from a key event

Description:

The **ShiftDown** convenience function returns true if **ShiftMask** is in the modifiers list, and false otherwise. This indicates if the Shift key was pressed when the key event was generated.

The **ControlDown** convenience function returns true if **ControlMask** is in the modifiers list, and false otherwise. This indicates if the Control key was pressed when the key event was generated.

2.7.3 XLookupString, NoSymbol**Types:**

```
val XLookupString: int -> Modifier list -> (string * int)
val NoSymbol: int
```

Syntax:

```
val (string, keysym) = XLookupString keycode modifiers ;
```

Arguments:

keycode Specifies the keycode from a key

Description:

modifiers Specifies the modifiers from a key ev

Description: the string for that combination the keysym for that combination

Description: **XLookupString** translates a key ev

Description: 8.2408 0 Td (to)Tj 13.1654 0 Td (a)Tj 9.31829 0 Td (KeySym)Tj 40.837 0 Td

ard interpretation

than the current X server time. Otherwise, the last-focus-change time is set to the specified time (**CurrentTime** is replaced by the current X server time). **XSetInputFocus** causes

error handling

Description:

The **XTranslateCoordinates** function takes coordinates from a source window's origin and returns these coordinates relative to a destination window's origin. If **XTranslateCoordinates** is called with destWindow equal to the source window, the coordinates will be returned relative to the source window's origin.

. **Fonts**

Types:

val CharBearing: **XCharStruct** **XCharStruct**

Window are on different screens or
a mapped child of destWindow with value
NoDrawable.

```
val FSAllCharsExist: XFontStruct -> bool
val FSDefaultChar: XFontStruct -> int
val FSMinBounds: XFontStruct -> XCharStruct
val FSMaxBounds: XFontStruct -> XCharStruct
val PSPerChar: XFontStruct -> XCharStruct list
val FSAscent: XFontStruct -> int
val FSDescent: XFontStruct -> int

val FSMinWidth: XFontStruct -> int
val FSMaxWidth: XFontStruct -> int
val FSMinHeight: XFontStruct -> int
val FSMaxHeight: XFontStruct -> int
```

Argument Typ

If the perChar list is empty, all glyphs between the first and last character indexes inclusive have the same information, as given by both minBounds and maxBounds.

Arguments:

- | | |
|-----------------|--|
| fs | Specifies the XFontStruct to use. |
| string | Specifies the character string. |
| bigChars | Specifies the character string as a list of char ^{int16_t} _{1992/4.0} float ^{float} _{1992/4.0} |

Description:


```
val MakeRect: XPoint -> XPoint -> XRectangle
val SplitRect: XRectangle -> (XPoint * XPoint)
```

Syntax:

```
val (topLeft,bottomRight) = SplitRect r ;
val r = MakeRect corner1 corner2 ;
```

Description:

MakeRect constructs an **XRectangle** given two points corresponding pair of opposite corners of the rectangle. This is useful when the order of the two points is not known, for example when dragging a rubber-banded box on the screen.

SplitRect returns the pair of points corresponding top-left and bottom-right corners of the **XRectangle**. It will always be the case that left <= right and top <= bottom .

2.9.6 NegativePoint

Types:

```
val NegativePoint: XPoint -> XPoint
```

Description:

NegativePoint negates both the x and y coordinates of the point. This is equivalent to reflecting about the x axis and the y axis.

2.9.7 OutsetRect, OffsetRect, IncludePoint

Types:

```
val OutsetRect: int -> XRectangle -> XRectangle
val OffsetRect: XRectangle -> XPoint -> XRectangle
val IncludePoint: XPoint -> XRectangle -> XRectangle
```

Description:

OutsetRect n R takes rectangle R and expands its area by n units in all directions.

Typically n is positive and this function is used to expand areas to include the same width all around. With a negative n, the area is contracted.

manufacturers' line drawing hardware, which may run many times

gc Speci es the GC

2.10.18 XSetState

Types:

```
val XSetState: GC -> int -> int -> GCFUNCTION -> int -> unit
```

Syntax:

```
XSetState gc foreground background function planeMask ;
```

Arguments:

background	Specifies the background pixel.	Sp
foreground	Specifies the foreground	

2.10.22 XSetTSOrigin

T

2.11.2 VisualRedMask, VisualGreenMask, VisualBlueMask

Types:

```
val VisualRedMask: Visual -> int
val VisualGreenMask: Visual -> int
val VisualBlueMask: Visual -> int
```

Syntax:

```
val redMask = VisualRedMask visual ;
```

bytesPerLine Sp

The

destArea Specifies coordinates relative to the origin of the dra

are returned for regions of the window that are obscured by other windows.

Description:

The **XInternAtom** function returns the atom identifier associated with the specified name string. If **onlyIfExists**

```
| PropertyVisual      of VisualList
| PropertyWindow      of DrawableList
| PropertyWMHints    of XWMHintList
| PropertyWMSizeHints of XWMSizeHintList
| PropertyWMIconSizes of (XRectangle *
|                      XRectangle *
|                      XRectangle) List
```

Properties:

WM_CLIENT_MACHINE	The string name of the machine on which the client application is running.
WM_COMMAND	The command and arguments, separated by ASCII n

perform the conversion then you call **XSendSelectionNotify** with property set to zero to indicate that the conversion failed. If the conversion was successful then the requestor will read the value from the property on the window, and will delete the property to indicate that the transfer has been completed.

.13 Scr n Sav r

2.13.1

XSetScreenSaver enables the screen saver. An interval of 0 disables the random-pattern motion. If no input from devices (keyboard, mouse, and so on) is generated for the specified number of timeout seconds once the screen saver is enabled, the screen saver is activated.

For each screen, if blanking is preferred and the hardware supports video blanking, the screen simply goes blank. Otherwise, if either exposures are allowed

©

Syntax:

```
val      default t      -> XGetDefault t      program      option      :
```

Arguments:

- option** Speci es the option name.
program Speci es the program name.

Description:

XGetDefault returns the value for the program and option entry in the user's defaults database. If **XGetDefault** fails then exception **XWindows** is raised with "XGetDefault failed".

.16 windows**2.16.1 XCreateWindow, XCreateSimpleWindow****Types:**

```
val      XCreateWindow: Drawable -> XPoint -> XRectangle ->
          int      -> int      -> WindowClass      -> Visual      ->
          XSetWindowAttributes      list      -> Drawable

val      XCreateSimpleWindow: Drawable -> XPoint -> XRectangle ->
          int      -> int      -> int      -> Drawable
```

Syntax:

```
val      window      = XCreateWindow: Drawable -> XPoint -> XRectangle ->
          parent      point      area
          borderWidth      depth      class      visual      attributes      :

val      window      = XCreateSimpleWindow: Drawable -> XPoint -> XRectangle ->
          parent      point      area
          borderWidth      borderPixel      backgroundPixel      :
```

Arguments:

- attributes** Speci es the initial values for the window's attributes.
backgroundPixel Speci es the bac


```
val XGetGeometry: Drawable -> (Drawable * XPoint * XRectangle * int * int)
val XGetWindowAttributes: Drawable -> XWindowAttributes
```

Syntax:

```
val (root, position, size, borderWidth, depth) = XGetGeometry w ;
val attributes = XGetWindowAttributes w ;
```

Arguments:

d Specifies the drawable, which can be a window

texID:IDrawable 37.0 Td (ot)Tj /R92 9.96264 Tf 80.3194 0 Td (Returns)Tj 37.8172 0 Td (the)Tj 17.1247 0

©

changes	Specifies a list of XWindowChanges .
w	Specifies the window to be reconfigured.
borderWidth	Specifies the width of the window border.
area	Specifies the interior dimensions of the window.
origin	Specifies the x and y coordinates, which define the new location of the top-left pixel of the window's border.

Syntax:

```
XRaiseWindow w ;
XLowerWindow w ;
X CirculateSubwindows w direction ;
X CirculateSubwindowsDown w ;
X CirculateSubwindowsUp w ;
XRestackWindows windows ;
```

Arguments:

direction	Specifies the direction (up or down) that you want to circulate the window. You can pass RaiseLowest or LowerHighest .
w	Specifies the window.
windows	Specifies the list of windows to be restacked.

Argument Type:

```
datatype CirculateDirection = RaiseLowest | LowerHighest
```

Description:

The **XRaiseWindow** function raises the specified window to the top of the stack so that no sibling window obscures it. If the windows are regarded

children are not affected. This is a convenience function equivalent to **XCirculateSubwindows** with **RaiseLowest** specified.

The **XCirculateSubwindowsDown** function lowers the highest mapped child of the specified window that partially or completely occludes another child. Completely

2.16.12 XUnmapWindow, XUnmapSubwindows

Types:

```
val XUnmapWindow: Drawable -> unit  
val XUnmapSubwindows: Drawable -> unit
```

Syntax:

```
XUnmapWindow w ;  
XUnmapSubwindows w ;
```

Arguments:

w Speci es the window.

Description:

The **XUnmapWindow** function unmaps the speci ed window and causes the X server to generate an **UnmapNotify** event. If the speci ed window is already

Syntax:

```
XSetWMClass w class ;  
val class = XGetWMClass w ;
```

Arguments:

class Speci es the class names for the window
w Speci es the window

Properties:

WM_CLASS Set by application programs to allow window and session managers to obtain the application's resources resource database.

_CLASS property on the speci ed window. **XGetWM-CLASS** returns the **WM_CLASS** property on the speci ed window.

2.17.4 XSetWMClientMachine, XGetWMClientMachine

Types:

```
val XSetWMClientMachine: Drawabl e
```

CLIENT

2.17.5 **XSetWMColormapWindows**, **XGetWMColormapWindows**

Types:

```
val XSetWMColormapWindows: Drawabl e
```

w Speci es the window.
commands Speci es the list of strings.

Description:

The **XSetWMCommand** function sets the WM_COMMAND property on the specified window. T

Description:

The **XSetWMIconName** convenience function performs a **X SetProperty** on the **WM_ICON_NAME** property. The **XSetWMIconName** function sets the name to be displayed in a window's icon.

The **XGetWMIconName** convenience function performs an **XGetTextProperty** on the **WM_ICON_NAME** property. The **XGetWMIconName** function returns the name to be displayed in the specified window's icon. If it succeeds, it returns the name, otherwise, if no icon name has been set for the window, it raises exception **XWindows** with "Xb

©

The **XGetWMSizeHints** function returns the

©

ha ter 5

E


```
ColormapNotify of { sendEvent: bool,  
                    window: Drawable,  
                    colormap: Colormap,  
                    new: bool,  
                    installed: bool }
```

Description:

The window member is set to the window whose associated colormap is changed, installed, or

3.7 onfi ur R qu st

Types:

```
datatype StackMode = Above | Below | TopIf | BottomIf | Opposite ;
```

```
ConfigureRequest of { sendEvent: bool,
                      parent: Drawable,
                      window: Drawable,
                      position: XPoint,
                      size: XRectangle,
                      borderWidth: int,
                      above: Drawable,
                      detail: StackMode }
```

borej b7.64729 0 Td (bsrecon reRed.j /A 1 Tf 0.1 0 0 -0.1 90

Description:

The parent member is set to the parent window. The window member is set to the window whose size, position, border width, and/or 7Tj 7j 2ar

3.17 MapNotify

Type: struct Hardware Ltd 1991,1994 X Reference 1.1

135

```
MapNotify of { sendEvent: bool,  
               event: Drawabl e,  
               wi ndow: Drawabl e,  
               overrideRedirect: bool }
```

Description:

The event member is set either to the window that was mapped or to its parent, depending on whether **StructureNotifyMask** or **SubstructureNotifyMask** was selected. The window member is set to the window that was mapped. The overrideRedirect member is set to the override-redirect attribute of the window. Window manager clients normally should ignore this window if the override-redirect attribute is true, because these events usually are generated from pop-ups, which override structure `cj 84.614.6056 0 Td (tol.5)Tj /A 1 Tf 0.1 0 0 -0.1`

```
ReparentNotify of { sendEvent:      bool ,
                    event:          Drawable,
                    windowparent:   Window, overrideRedirect: bool }
```

Description: The event member is set either to the reparented window or to the old or the new parent, depending on whether **StructureNotifyMask** or **SubstructureNotifyMask** was selected.

The window member is set to the window that was reparented. The parent member is set to the new parent window. The position member is set to the reparented window's coordinates relative to the new parent window's origin and defines the upper-left outer corner of the reparented window. The overrideRedirect member is set to the override-redirect attribute of the window specified by the window member. Window manager clients normally should ignore this window if the overrideRedirect member is true.

3. 1 ResizeRequest

Types:

```
ResizeRequest of { sendEvent: bool ,
                   window:   Drawable,
                   size:     XRectangle }
```

The window member is set to the window whose size another client attempted to change.

Description: The size member is set to the inside size of the window, excluding the border.

3. SelectionClear

Types:

```
SelectionClear of { sendEvent: bool,
                     window:   Drawable,
                     selection: Atom,
                     time:    Time }
```

The window member is set to the window losing ownership of the selection.

Description: The selection member is set to the selection atom. The time member is set to the last change time recorded for the selection. The owner member is the window that was specified by the current owner in its **XSetSelectionOwner** call.

3. 3 SelectionNotify

Types:

```
SelectionNotify of { sendEvent: bool,  
                    requestor: Drawable,  
                    selection: int,  
                    target: int,  
                    property: int,  
                    time: int }
```

Description:

The requestor member is set to the window associated with the

Description:

The event member is set either to the unmapped window or to its parent, depending on whether **StructureNotifyMask** or **SubstructureNotifyMask** was selected. This is the window used by the X server to report the event. Th

ha ter 4

Fr ■ l Err r

4.1 **BadPixmap**

Description:

A value for a Pixmap does not name a defined Pixmap. ML type-checking should avoid this error.

4.13 **BadRequest**

Description:

This should never occur in Xlib since only standard requests are made.

4.14 **BadValue**

Description:

Some numeric value falls outside the range of values accepted by the request.

XAllocColorCells	Number of colours must be positive and planes must be non-negative.
XAllocColorPlanes	Number of colours must be positive, and reds, green and blues must be non-negative
XFreeColors	Specified pixel is not a valid index into the colormap.
XBell	Percent must be ~100 to 100.
XResizeWindow	Window width must be non-zero.
XCopyPlane	Plane must have one bit set to 1, and specify an existing plane.
XCreateGlyphCursor	Source char and mask char must exist in the font.
XSetDashes	Dash elements must be positive and less than 256.
XCreatePixmap	Specified width must be non-zero, and depth must be supported.
xSetScreenSav	arealess

W8Td (XSetDashes)Tjtor/R92 9.9

Ind x

Above	107, 130	BitmapOpenFailed	97	
AboveOf	66, 67	BitmapPad	31	
AddPoint	BitmapSuccess	97		
AllocAll	BitmapUnit	32		
AllocNone	BlackPixel	15, 16		
AllocType	Blanking	9		
AllowExp	Bottom	67, 68		
AllPlanes	BottomIf	107, 130		
Always	BottomLeft	67, 68		
And	BottomRight	67, 68		
AnyButton	Button1	126, 126, 127		
AnyModi	Button1Mask	126, 127, 132		
ArcChord	Button1MotionMask	54		
ArcPieSlic	Button2	126, 127, 132		
Area	Button2Mask	46, 49, 51, 67, 68, 126, 127, 132		
	Button2MotionMask	54		
	Button3	126, 127		
BackingStore	Button3Mask	126, 127, 132		
BadAccess	Button3MotionMask	54		
BadAlloc	Button4	126, 127, 132		
BadAtom	Button4Mask	126, 127, 132		
BadColor	Button4MotionMask	54		
BadCursor	Button5	126, 127, 132		
BadDrawable	Button5Mask	126, 127, 132		
BadFont	Button5MotionMask	54		
BadGC	ButtonClick	49, 125, 126, 127		
BadIDChoice	ButtonClickMask	54		
BadImplementation	ButtonMotionMask	54		
BadLength	ButtonName	126		
BadMatch	ButtonPress	140		
	24, 25, 29, 40, 41, 42, 57, 74,			
	75, 76, 78, 83, 84, 89, 97, 100, 101,			
	106, 107, 112, 141			
BadPixmap		142		
BadRequest		142		
BadValue	18, 19, 22, 29, 42, 75, 79, 9 (84.)Tj	15.9354 0 T96,		
	98, 107, 108, 142			
BadWindow		142		
Below	107, 130			
BelowOf	66, 67			
BitmapBitOrder		31		
BitmapFileInvalid		97		
BitmapNoMemory		97		

CharAscent	59, 60
CharAttributes	59
CharDescent	59, 60
CharLBearing	59
CharRBearing	59
CharWidth	59, 60
CirculateDirection	110, 111
CirculateNotify	111, 125, 128
CirculateRequest	54, 111, 125, 128
ClipByChildren	40, 75, 84
ColormapChangeMask	54
ColormapExists	33
ColormapID	33
ColormapNotify	25, 26, 105, 125, 128
Complex	49, 50
ConfigureNotify	125, 129
ConfigureRequest	54, 108, 111, 112, 125, 130
ControlDown	52, 53
ControlMask	53, 126, 127, 132
Convex	49, 50
CoordMo	

FontDirection	59, 60, 61, 64
FontExists	14, 33
FontID	33
FontLeftToRight	61, 62, 65
FontRightToLeft	61, 62, 65
ForgetGravity	102,
FSAllCharsExist	59
FSAscent	59
FSDefaultChar	59
FSDescent	59
FSDirection	59
FSFont	59
FSMaxBounds	59, 60
FSMaxByte1	59
FSMaxChar	59
FSMaxHeight	59, 60
FSMaxWidth	59, 60
FSMinBounds	59, 60
FSMinByte1	59
FSMinChar	59
FSMinHeight	59, 60
FSMinWidth	59, 60



GCArcMode	71, 75, 76
GCBackground	71
GCCapStyle	71, 73, 82
GCClipMask	71
GCClipOrigin	71
GCDashList	71
GCDashO_set	71
GCExists	14, 33
GCFillRule	71, 75, 79, 80
GCFillStyle	71, 74, 80
GCFont	71
GCForeground	71
GCFunction	71, 81, 83
GCGraphicsExposures	71
GCID	33
GCJoinStyle	71, 73, 82
GCLineStyle	71, 73, 82
GCLineWidth	71
GOrder	77, 78dth

71,71yle

IsUnviewable	102, 103
IsViewable	102, 103

J

JoinBevel	73, 82
JoinMiter	11, 73, 82
JoinRound	73, 82

K

KeymapNotify	125, 134
KeymapStateMask	54
KeyPress	125, 126, 127
KeyPressMask	54
KeyRelease	125, 126, 127
KeyReleaseMask	54

L

LeaveNotify	55, 125, 131, 132, 133
LeaveWindowMask	54
Left	67, 68
LeftOf	66, 67
LineDoubleDash	73, 74, 82
LineOnO_Dash	73, 74, 82
LineSolid	73, 74, 82
LockMask	126, 127, 132
LowerHighest	111, 112
LSBFIRST	31, 32, 85, 87

M

MakeRect	68, 69
MapNotify	109, 125, 135
MapRequest	54, 109, 125, 135
MapState	102
MaxCmapsOfScreen	37
Message	56, 125, 135
MinCmapsOfScreen	36
Mod1Mask	126, 127, 132
Mod2Mask	126, 127, 132
Mod3Mask	126, 127, 132
Mod4Mask	126, 127, 132
Mod5Mask	126, 127, 132

Modi 140017866436 0 Td (52,)Tj 16.0488132, d13(53,)Tj 16.0204 0 Td (109,)Tj 21.0018 0 Td (126,)Tj 21.0018 0 Td

PointerWindow	37
PolyShape	49
PPosition	121, 122
PreferBlanking	94
PResizeInc	121, 122

VisualBlueMask	86
VisualClass	23, 24
VisualExists	33
VisualGreenMask	86
VisualID	33
VisualRedMask	86

W

WestGravity	102, 103
WhenMapped	35, 100, 102, 103, 105
WhitePixel	15, 16
Width	67, 68
WindingRule	75, 80
WindowClass	99, 100, 102
Within	66, 67
WM_CLASS	115, 120
WM_CLIENT_MACHINE	92, 115, 120
WM_COLORMAP_WINDOWS	116
WM_COMMAND	92, 117, 120
WM_HINTS	118, 120
WM_ICON_NAME	92, 119, 120
WM_NAME	92, 119, 120
WM_NORMAL_HINTS	120, 122, 123
WM_SIZE_HINTS	121, 122
WM_TRANSIENT_FOR	9114

X

XActivateScreenSaver	94, 95
XAddPixel	86, 88
XAllocColor	17, 18, 19, 20, 25
XAllocColorCells	17, 18, 19, 25, 142
XAllocColorPlanes	17, 18, 19, 25, 27, 142
XAllocNamedColor	17, 18, 19, 20, 25
XArc	42, 43, 49, 91
XAutoRepeatOff	98
XAutoRepeatOn	98
XA_PIXMAP	137

XA_PRIM62 30.0471 4.25195E+92 j q-10 Tj 0 6 Td (6HTNTJS)Tj 131092625095dY(1620,)Tj 21.001824.25195 rY16.04 4.6m

XGetAtomName	90, 91
XGetDefault	20, 98, 99
XGetFontPath	64
XGetGeometry	101, 102, 104
XGetIconSizes	113, 114
XGetImage	88, 89, 141
XGetInputFocus	56, 57
XGetPixel	86, 88
XGetRGBColormaps	26, 28
XGetScreenSaver	94, 95
XGetSelectionOwner	92, 93
XGetSubImage	88, 89, 141
XGetTextProperty	91, 92, 115, 119
XGetTransientForHint	114
XGetWindowAttributes	101, 102, 104
XGetWindowBorderWidth	104
XGetWindowChildren	104
XGetWindowDepth	104
XGetWindowParent	104
XGetWindowPosition	104
XGetWindowRoot	104
XGetWindowSize	104
XGetWMClass	114, 115
XGetWMClientMachine	115
XGetWMColormapWindows	116
XGetWMCommand	116, 117
XGetWMHints	117, 118
XGetWMIconName	118, 119
XGetWMName	119
XGetWMNormalHints	121, 122
XGetWMSizeHints	121, 122
XImage	85, 86, 87, 88, 89
XInstallColormap	25, 26, .82
XInternAtom	90, 91
XListFonts	60, 61
XListFontsWithInfo	60, 61
XListInstalledColormaps	25, 26
XLoadFont	61, 63
XLoadQueryFont	11, 61, 63
XLookupColor	19, 20
XLookupString	53
XLowerWindow	110, 111
XMapRaised	108, 109

XSetWindowColormap 11, 24, 25, 26, 141
XSetWMClass 114, 115, 120
XSetWMClientMachine 115
XSetWMColormapWindows 116
XSetWMCommand 116, 117, 120
XSetWMHints 114, 117, 118, 120
XSetWMIconName 118, 119, 120
XSetWMName 119, 120
XSetWMNormalHints 120, 121, 122
XSetWMProperties 119, 120
XSetWMSizeHints 121
XStandardColormap 26, 27, 91