OpenWindows Source Code C

Introduction

This appendix contains sample source code for OLIT and the OpenWindows Version 4 server, including a tutorial on how to internationalize an OLIT program, text buffer utility functions for internationalized OLIT, man pages for new OLIT widgets, sample source for the DDX-based CG3 framebuffer, and the sample tablet device driver using the dynamically-loadable mechanism and X input extension.

Internationalization Tutorial

The following session shows you how to internationalize a program. The example shown pops up the following window:

Code Example 16-5 memo.c (not internationalized)

```
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xol/OpenLook.h>
#include <Xol/OblongButt.h>

main(argc, argv)
  int   argc;
  char *argv[];
{
  Widget     toplevel, msg_widget;
  void       quit();
  XtAppContext app;
```

Code Example 16-5 memo.c (not internationalized) (Continued)

```
#include <X11/Intrinsic.h>
  /* Initialize the Intrinsics */
  OlToolkitInitialize((XtPointer)NULL);
  toplevel = XtAppInitialize(&app, "Memo", (XrmOptionDescList)NULL, 0,
                          (Cardinal *)&argc, argv, NULL, (ArgList) NULL, 0);
  /* Create the OblongButton widget. */
  msg_widget = XtVaCreateManagedWidget("msg", oblongButtonWidgetClass,
                                      toplevel, XtNlabel, "Press to quit",
                                      NULL);
  /* Register the callback to be called when the button is pressed */
  XtAddCallback(msg_widget, XtNselect, quit, NULL);

  /* Realize the widgets and enter the event loop. */
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app);
}
```

Internationalization Techniques

The following steps are suggested for internationalizing a program.

1. **Set a text format for each widget.**
2. **Remove programmatic setting of strings, labels, and messages from the source code.**
3. **Put displayed text in resource defaults files.**
4. **Make sure you are using the right types:**

`String`	➡	`OlStr`
`XFontStruct *`	➡	`OlFont`

The following is the output from the internationalized version of `memo.c`:

***msg.label: Language Specific Label**

Put this specification in a resource defaults file

Code Example 16-6   Internationalized memo.c

```
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xol/OpenLook.h>
#include <Xol/OblongButt.h>

main(argc, argv)
  int   argc;
  char *argv[];
{
  Widget     toplevel, msg_widget;
  void       quit();
  XtAppContext app;

  OlToolkitInitialize((XtPointer)NULL);
  OlSetDefaultTextFormat(OL_MB_STR_REP);
  toplevel = XtAppInitialize(&app, "Memo",
              (XrmOptionDescList)NULL, 0,
              (Cardinal *)&argc, argv, NULL, (ArgList) NULL, 0);
  /* Create the OblongButton widget. */
  msg_widget = XtVaCreateManagedWidget("msg", oblongButtonWidgetClass,
                           toplevel, XtNlabel, "Press to quit",
                           NULL);
  /* Register the callback to be called when the button is pressed */
  XtAddCallback(msg_widget, XtNselect, quit, NULL);

  /* Realize the widgets and enter the event loop. */
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app);
}
```

Remove resoures set programmatically

Resource Defaults Files

When internationalizing an application, the primary focus involves moving text from your source code to resource files. The following section deals with initialization of resources.

1. **Read during initialization (`XtAppInitialize`).**
2. **Files are loaded in this order:**
   - application defaults file
     `$XFILESEARCHPATH`
   - per user application defaults file
     `$XUSERFILESEARCHPATH`
   - application user's defaults
     `RESOURCE_MANAGER or .Xdefaults`
   - application user's per-host defaults
     `$XENVIRONMENT`
   - command-line arguments

XFILESEARCHPATH

The `XFILESEARCHPATH` environment variable allows you to list your resource files so the X resource manager can read them when initializing an application. The `XFILESEARCHPATH` environment variable consists of the following:

- A list of colon-separated paths that include variables.
- The variables are expanded at run time.

A default `XFILESEARCHPATH` may look like this:

```
XFILESEARCHPATH=\
      /usr/openwin/lib/locale/%L/%T/%N%S:\
      /usr/openwin/lib/%T/%N%S
```

You can add additional paths to `XFILESEARCHPATH` by using something similar to the following lines:

```
XFILESEARCHPATH=\
       /usr/openwin/lib/locale/%L/%T/%N%S:\
       /usr/openwin/lib/%T/%N%S:\
       /home/john/%T/%L/%N%S
```

The variables are expanded as follows:

%T	type	(app-defaults)
%N	name	(class name)

%S	suffix	(ignored on UNIX systems)
%L	lang	(locale)

For example, if `XFILESEARCPATH` includes

```
/home/john/%T/%L/%N%S
```

and your locale is set to Japanese, as in

```
memo -xnllanguage ja
```

and your class name of application is “Memo”, as in

```
XtAppInitialize(app, "Memo", ... )
```

your path will expand to

```
/home/john/app-defaults/ja/Memo
```

If you use the previous example, but instead set the locale to Korean, path will expand to

```
/home/john/app-defaults/ko/Memo
```

The following program demonstrates how to use resources to internationalize a program. The example used is called `fruits.c`.

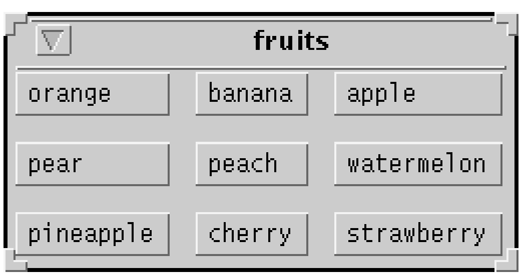
Code Example 16-7 fruits.c Program

```
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xol/OpenLook.h>
#include <Xol/Nonexclusi.h>
#include <Xol/RectButton.h>
#include <stdio.h>

static void select_callback();
static char *names[] = { "orange", "banana", "apple", "pear", "peach",
                         "watermelon", "pineapple", "cherry", "strawberry" };

void main(argc, argv)
  int   argc;
  char *argv[];
{
  XtAppContext app;
  Widget       toplevel, container, button[9];
  int i;
```

Code Example 16-7 fruits.c Program (Continued)

```
    /*
     * Initialize OLIT and the Intrinsics
     */
    OlToolkitInitialize((XtPointer)NULL);
    OlSetDefaultTextFormat(OL_MB_STR_REP);
    toplevel = XtAppInitialize(&app, "Fruits", NULL, 0, &argc,
                               argv, NULL, NULL, 0);
  /*
     * Create the OblongButton widget.
     */
    container = XtVaCreateManagedWidget("container", nonexclusivesWidgetClass,
                                        toplevel,
                                        XtNlayoutType, OL_FIXEDCOLS,
                                        XtNmeasure,    3,
                                        NULL);
    for(i=0;i<XtNumber(names);i++)
        button[i] = XtVaCreateManagedWidget(names[i], rectButtonWidgetClass,
                                        container, NULL);

    XtRealizeWidget(toplevel);
    XtAppMainLoop(app);
}
```

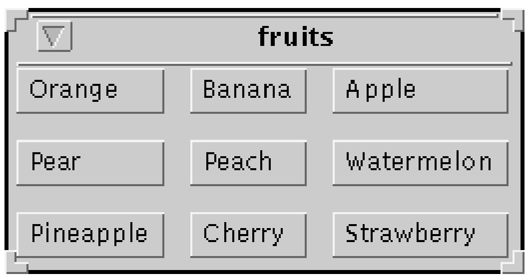
Using default resource, the output looks like this:



With C locale resources set in your defaults file, such as

```
/home/john/app-defaults/Fruits
```

your output will look like this:




```
*orange.label:      Orange
*banana.label:      Banana
*apple.label:       Apple
*pear.label:        Pear
*peach.label:       Peach
*watermelon.label:  Watermelon
*pineapple.label:   Pineapple
*cherry.label:      Cherry
*strawberry.label:  Strawberry
```

With Korean language resources set, such as

```
/home/john/app-defaults/ko/Fruits
```

your output will look like this:


```
*orange.label:      \077\100\067\073\101\166
*banana.label:      \071\131\063\052\063\052
*apple.label:       \073\147\060\172
*pear.label:        \071\150
*peach.label:       \072\071\074\176\076\106
*watermelon.label:  \074\166\071\132
*pineapple.label:   \106\104\100\116\076\126\107\103
*cherry.label:      \076\136\065\116
*strawberry.label:  \065\176\061\142
```

Note – The octal characters are displayed in the text representation because Korean locale-specific characters are not part of the ASCII character set. To display the octal strings as Korean characters, use the OpenWindows text editor with the Korean localization package.

We just saw how to internationalize OLIT applications. An important part of the application is the supporting help document. Below, we demonstrate how to access help in an internationalized environment.

The Help Mechanism

The variable %T in XFILESEARCHPATH's search path expands to the location of the help file.

For example, if XFILESEARCPATH includes

```
/home/john/%T/%L/%N%S
```

and the locale is set to Japanese

```
memo -xnllanguage ja
```

and DISK_SOURCE is set to Help file: HelpFile, as in

```
OlRegisterHelp(OL_WIDGET_HELP, wid, title,
                       OL_DISK_SOURCE, "HelpInfo");
```

XFILESEARCHPATH expands to

```
/home/john/help/ja/HelpInfo
```

The following program shows how to register help in internationalized-ready programs.

Code Example 16-8 help.c

```
/* Header Files Deleted */
static OlStr HelpTitle = "OblongButton";

static void button_callback(Widget, XtPointer, XtPointer);

void
main(unsigned int argc, char **argv)
{
  XtAppContext app;
  Widget       toplevel, con, but1;

  OlToolkitInitialize((XtPointer)NULL);
  toplevel = XtAppInitialize(&app, "Help", NULL, 0, &argc,
                             argv, NULL, NULL, 0);
  con = XtCreateManagedWidget("control", controlAreaWidgetClass,
                              toplevel, NULL, 0);
  /*
   * Create the OblongButton widget.
   */
  but1 = XtVaCreateManagedWidget("button", oblongButtonWidgetClass,
                                 con, NULL);
  XtAddCallback(but1, XtNselect, button_callback, NULL);
  OlRegisterHelp(OL_WIDGET_HELP, (XtPointer)con, HelpTitle,
```

Code Example 16-8 help.c (Continued)

```
                 OL_DISK_SOURCE, (XtPointer)"HelpFile");
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app);
}
```

Text Buffer Utility Functions for Internationalization in OLIT

The following functions are new TextEdit Utility Functions.

OlTextEditOlTextBuffer()

Synopsis

```
#include <buffutil.h>
#include <Oltextbuff.h>
#include <Dynamic.h>
#include <TextEdit.h>
  ...
OlTextBufferPtr OlTextEditOlTextBuffer(TextEditWidget ctx);
```

Arguments

`ctx`	The TextEdit widget to retrieve the `OlTextBufferPtr` for

Description

`OlTextEditOlTextBuffer()` function retrieves the `OlTextBufferPtr` associated with the TextEdit widget `ctx`. This buffer exists only when the value of `XtNtextFormat` for the widget is not `OL_SB_STR_REP`. This pointer can be used to access the facilities provided by the multibyte functions. For a description of these functions, see the following section.

In case, `XtNtextFormat` is `OL_SB_STR_REP`, this function returns a `NULL` pointer.

The following functions are new Text Buffer Utility Functions.

The text buffer functions in the sections below have been added to provide multibyte equivalents to the single-byte OLIT 3.1 text buffer functions.

OlAllocateTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
OlTextBufferPtr OlAllocateTextBuffer(OlStrRep strrep,
  char* filename,
  TextUpdateFunction update_func,
  XtPointer data);
```

Arguments

`strrep`	Specifies the text format
`filename`	Specifies the filename
`update_func`	The update function
`data`	Client data

Description

The `OlAllocateTextBuffer()` function is used to allocate a new `OlTextBuffer`. After it allocates the structure, it initializes the members of the structure, allocating storage, setting initial values. It also registers the update function provided by the caller. This function does not normally need to be called by an application developer since the `OlReadFileIntoTextBuffer()` and `OlReadStringIntoTextBuffer()` functions call this routine before starting their operation. The routine returns a pointer to the allocated `OlTextBuffer`. The `OlFreeTextBuffer()` function should be used to deallocate the storage allocated by this routine.

The `strrep` argument can have the following values:

Value	Meaning
`SB_STR_REP`	Single-byte character representation
`WC_STR_REP`	Wide character representation
`MB_STR_REP`	Multi-byte character representation

See Also

`OlFreeTextBuffer()`, `OlReadFileIntoTextBuffer()`, `OlReadStringIntoTextBuffer()`

OlBackwardScanTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
ScanResult OlBackwardScanTextBuffer(OlTextBufferPtr text,
  OlStr exp,
  TextLocation* location);
```

Arguments

`text`	The text buffer
`exp`	The expression to scan for
`location`	The location to start scanning at

Description

The `OlBackwardScanTextBuffer()` function is used to scan towards the beginning of the buffer, for a given expression in the `OlTextBuffer` starting at `location`. The `ScanResult` can have the following values :

ScanResult	Meaning
SCAN_NOTFOUND	The scan wrapped without finding a match.
SCAN_WRAPPED	A match was found at a location after the start location.
SCAN_FOUND	A match was found at a location before the start location.
SCAN_INVALID	Either the location or the expression was invalid.

See Also

`OlForwardScanTextBuffer()`

OlCopyTextBufferBlock()

Synopsis

```
#include <Oltextbuff.h>
```

```
...
int OlCopyTextBufferBlock(OlTextBufferPtr text,
  OlStr outbuffer,
  int num_bytes,
  TextPosition start_position,
  TextPosition end_position);
```

Arguments

`text`	The text buffer
`outbuffer`	The buffer to output the text to
`num_bytes`	Size of the text block
`start_position`	Beginning of text block
`end_position`	End of text block

Description

The `OlCopyTextBufferBlock()` function is used to retrieve a text block from the `OlTextBuffer`. The block is defined as the characters between `start_position` and `end_position` inclusive. If `num_bytes` is not sufficient, `OlCopyTextBufferBlock()` returns `-1` otherwise it returns actual bytes used.

Note – The storage for the copy is allocated by the caller. It is the responsibility of the caller to ensure that enough storage is allocated to copy (`end_position` - `start_position`) + (bytes to store `NULL` character).

See Also

`OlAllocateTextBuffer()`, `OlRegisterTextBufferUpdate()`, `OlGetTextBufferCharAtLoc()`, `OlGetTextBufferLine()`

OlEndCurrentTextBufferWord()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlEndCurrentTextBufferWord(OlTextBufferPtr text,
  TextLocation* current);
```

Arguments

`text`	The text buffer
`current`	Specifies current location

Description

The `OlEndCurrentTextBufferWord()` function is used to locate the end of a word in the `OlTextBuffer` relative to a given current location. It returns the location of the end of the current word. The return value will equal the given current value if the current location is already at the end of a word. If the location is not in a word, it returns the end of the *"not word"* region it is in.

Note – The location passed to `OlEndCurrentTextBufferWord()` is modified. It contains the end of the current buffer word (or"*not word*") at the end of the call.

See Also

`OlPreviousTextBufferWord(),OlNextTextBufferWord()`

OlForwardScanTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
ScanResult OlForwardScanTextBuffer(OlTextBufferPtr text,
  OlStr exp,
  TextLocation* location);
```

Arguments

`text`	The text buffer
`exp`	Specifies the expression to scan for
`location`	The location to start scanning at

Description

The `OlForwardScanTextBuffer()` function is used to scan towards the end of the buffer for a given expression in the `OlTextBuffer` starting at `location`. A `ScanResult` is returned which indicates:

ScanResult	Meaning
`SCAN_NOTFOUND`	The scan wrapped without finding a match.
`SCAN_WRAPPED`	A match was found at a location before the start location.
`SCAN_FOUND`	A match was found at a location after the start location.
`SCAN_INVALID`	Either the location or the expression was invalid.

See Also

`OlBackwardScanTextBuffer()`

OlFreeTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
void OlFreeTextBuffer(OlTextBufferPtr text,
  TextUpdateFunction update_func,
  XtPointer data);
```

Arguments

`text`	The text buffer to free
`exp`	The update function
`data`	Data

Description

The `OlFreeTextBuffer()` procedure is used to deallocate storage associated with a given `OlTextBuffer`.

Note – The storage is not actually freed if the `OlTextBuffer` is still associated with other update function/data pairs.

See Also

`OlAllocateTextBuffer(), OlRegisterTextBufferUpdate()`

OlGetTextBufferBlock()

Synopsis

```
#include <Oltextbuff.h>
...
OlStr OlGetTextBufferBlock(OlTextBufferPtr text,
  TextLocation* start_location,
  TextLocation* end_location);
```

Arguments

`text`	The text buffer to retrieve block from
`start_location`	Start of the text block
`end_location`	End of the text block

Description

The `OlGetTextBufferBlock()` function is used to retrieve a text block from the `text` `TextBuffer`. The block is defined as the characters between `start_location` and `end_location` inclusive. It returns a pointer to a string containing the copy. If the parameters are invalid, `NULL` is returned.

Note – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

See Also

`OlAllocateTextBuffer(),OlRegisterTextBufferUpdate(), OlGetTextBufferCharAtLoc(),OlGetTextBufferLine()`

OlGetTextBufferBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
Buffer* OlGetTextBufferBuffer(OlTextBufferPtr text,
  TextLine line);
```

Arguments

`text`	The text buffer
`line`	The line to retrieve a pointer for

Description

The `OlGetTextBufferBuffer()` function is used to retrieve a pointer to the `Buffer` stored in `OlTextBuffer` text for a line. This pointer is volatile; subsequent calls to any `OlTextBuffer` routine may make it invalid. If a more permanent copy of this `Buffer` is required, the buffer utility, `CopyBuffer()` can be used to create a private copy of it.

See Also

`OlGetTextBufferBlock(),OlAllocateTextBuffer(), OlRegisterTextBufferUpdate()`

OlGetTextBufferCharAtLoc()

Synopsis

```
#include <Oltextbuff.h>
...
OlStr OlGetTextBufferCharAtLoc(OlTextBufferPtr text,
  TextLocation* location);
```

Arguments

`text`	The text buffer
`location`	The location at which to get a character

Description

The `OlGetTextBufferCharAtLoc()` function is used to retrieve a character stored in the `OlTextBuffer` at `location`. It returns either the pointer to the character itself or `NULL` if `location` is outside the range of valid locations within the `OlTextBuffer`.

See Also

`OlAllocateTextBuffer()`, `OlRegisterTextBufferUpdate()`, `OlGetTextBufferBlock()`, `OlGetTextBufferLine()`

OlGetTextBufferFileName()

Synopsis

```
#include <Oltextbuff.h>
...
String OlGetTextBufferFileName(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer to get a file name for

Description

The function, `OlGetTextBufferFileName()`, returns the file name associated with the buffer. Otherwise, it returns `NULL`.

OlGetTextBufferLine()

Synopsis

```
#include <Oltextbuff.h>
...
OlStr OlGetTextBufferLine(OlTextBufferPtr text,
  TextLine lineindex);
```

Arguments

`text`	The text buffer
`lineindex`	Index of the line to retrieve the contents for

Description

The `OlGetTextBufferLine()` function is used to retrieve the contents of line from the `OlTextBuffer`. It returns a pointer to a string containing the copy of the contents of the line or `NULL` if the line is outside the range of valid lines in text.

Note – The storage for the copy is allocated by this routine. It is the responsibility of the caller to free this storage when it becomes dispensable.

See Also

`OlAllocateTextBuffer()`, `OlRegisterTextBufferUpdate()`, `OlGetTextBufferCharAtLoc()`, `OlGetTextBufferBlock()`

OlGetTextUndoDeleteItem()

Synopsis

```
#include <Oltextbuff.h>
...
OlTextUndoItem OlGetTextUndoDeleteItem(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer to get the undo delete item for

Description

The function, `OlGetTextUndoDeleteItem()`, returns an `OlTextUndoItem` structure containing the value of a “deleted” undo item. A copy of the deleted string is provided in the returned structure.

OlGetTextUndoInsertItem()

Synopsis

```
#include <Oltextbuff.h>
...
OlTextUndoItem OlGetTextUndoInsertItem(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer to get the undo insert item for

Description

The function, `OlGetTextUndoInsertItem()`, returns an `OlTextUndoItem` structure containing the value of an “insert” undo item. A copy of the insert string is provided in the returned structure.

OlIncrementTextBufferLocation()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlIncrementTextBufferLocation(OlTextBufferPtr text,
  TextLocation* location,
  TextLine line,
  TextPosition offset);
```

Arguments

`text`	The text buffer
`location`	The location to increment
`line`	The lines to increment by
`offset`	The character offset to increment by

Description

The `IncrementTextBufferLocation()` function is used to increment a location by either line and/or offset characters. It returns a pointer to the modified starting location. If `line` or `offset` are negative, the function performs a decrement operation. If the starting location is `NULL`, `NULL` is returned. If the starting location or the resulting location is invalid, the pointer to the starting location is returned without modification; otherwise the starting `location` is modified and a pointer to the starting `location` is returned.

Note – Do not expect the location passed to this function to remain unchanged.

See Also

`OlNextLocation(),OlPreviousLocation()`

OlIsTextBufferEmpty()

Synopsis

```
#include <Oltextbuff.h>
...
Boolean OlIsTextBufferEmpty(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer to test

Description

The function, `OlIsTextBufferEmpty()`, returns `True` if the text buffer is empty; otherwise it returns `False`.

OlIsTextBufferModified()

Synopsis

```
#include <Oltextbuff.h>
...
Boolean OlIsTextBufferModified(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer

Description

The function, `OlIsTextBufferModified()`, returns `True` if the text buffer has been modified since last save; otherwise it returns `False`.

OlLastCharInTextBufferLine()

Synopsis

```
#include <Oltextbuff.h>
...
```

```
int OlLastCharInTextBufferLine(OlTextBufferPtr text,
  TextLine line);
```

Arguments

`text`	The text buffer
`line`	The line to find the last character offset for

Description

The function, `OlLastCharInTextBufferLine()`, returns the character offset of the last character in the text buffer line.

OlLastTextBufferLine()

Synopsis

```
#include <Oltextbuff.h>
...
TextLine OlLastTextBufferLine(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer to find the last line number for

Description

The function, `OlLastTextBufferLine()`, returns the last text buffer line number.

OlLastTextBufferLocation()

Synopsis

```
#include <Oltextbuff.h>
...
TextPosition OlLastTextBufferLocation(OlTextBufferPtr text);
                         TextLocation* last);
```

Arguments

`text`	The text buffer
`last`	Address to return the last location to

Description

The `OlLastTextBufferLocation()` function returns the pointer to the last valid `TextLocation` in the `OlTextBuffer` associated with text. If the second argument is `NULL`, space for last `TextLocation` is allocated, otherwise the second argument contains the last valid `TextLocation`.

See Also

`OlLastTextBufferPosition(), OlFirstTextBufferLocation()`

OlLastTextBufferPosition()

Synopsis

```
#include <Oltextbuff.h>
...
TextPosition OlLastTextBufferPosition(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer

Description

The `OlLastTextBufferPosition()` function returns the last valid `TextPositon` in the `OlTextBuffer` associated with text.

See Also

`OlLastTextBufferLocation(), OlFirstTextBufferLocation()`

OlLineOfPosition()

Synopsis

```
#include <Oltextbuff.h>
...
TextLine OlLineOfPosition(OlTextBufferPtr text,
  TextPosition position);
```

Arguments

`text`	The text buffer

Description

The `OlLineOfPosition()` function is used to translate a position in the `OlTextBuffer` to a line index. It returns the translated line index or `EOF` if the position is invalid.

See Also

`OlLineOfPositon(),OlPositionOfLocation(),`
`OlLocationOfPosition()`

OlLinesInTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
int OlLinesInTextBuffer(OlTextBufferPtr text);
```

Arguments

`text`	The text buffer to get the number of lines for

Description

The function, `OlLinesInTextBuffer()`, returns the number of lines in the given `OlTextBuffer`.

OlLocationOfPosition()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlLocationOfPosition(OlTextBufferPtr text,
  TextPosition position,
  TextLocation* location);
```

Arguments

`text`	The text buffer
`position`	The position to translate to a TextLocation
`location`	The address to store the translated TextLocation

Description

The `OlLocationOfPosition()` function is used to translate a position in the `OlTextBuffer` to a `TextLocation`. It expects a pointer to a `TextLocation` in the third argument to deposit the translated `TextLocation`. If the third argument is `NULL`, it allocates space. It returns a pointer to the allocated `TextLocation` or the passed `TextLocation`, with the translated value deposited in it. If the position is invalid, the `Buffer` pointer buffer is set to `NULL` and the line and offset members are set to the last valid location in the `OlTextBuffer`; otherwise buffer is set to a non-NULL (though useless) value.

Note – The storage space for `TextLocation`, if not provided by the caller, is allocated by this function.

See Also

`OlLineOfPosition()`, `OlPositionOfLocation()`, `OlLocationOfPosition()`

OlNextLocation()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlNextLocation(OlTextBufferPtr text,
  TextLocation* current);
```

Arguments

`text`	The text buffer
`current`	The current location

Description

The `OlNextLocation()` function returns the pointer to the `TextLocation` which follows the given current location in a `OlTextBuffer`. If the current location points to the end of the `OlTextBuffer`, this function wraps.

Note – The location passed to this function is modified. It contains the next location at the end of the call.

See Also

`OlPreviousLocation()`

OlNextTextBufferWord()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlNextTextBufferWord(OlTextBufferPtr text,
  TextLocation* current);
```

Arguments

`text`	The text buffer
`current`	The current location

Description

The `OlNextTextBufferWord()` function is used to locate the beginning of the next word from a given current location in a `OlTextBuffer`. If the current location is within the last word in the `OlTextBuffer`, the function wraps to the beginning of the `OlTextBuffer`.

Note – The location passed to this function is modified. It contains the start of the next buffer word at the end of the call.

See Also

`OlPreviousTextBufferWord(), OlStartCurrentTextBufferWord()`

OlNumBytesInTextBufferLine()

Synopsis

```
#include <Oltextbuff.h>
...
int OlNumBytesInTextBufferLine(OlTextBufferPtr text,
  TextLine line);
```

Arguments

`text`	The text buffer
`line`	The line to get the number of bytes for

Description

The function `OlNumBytesInTextBufferLine()` returns the number of bytes in line.

OlNumCharsInTextBufferLine()

Synopsis

```
#include <Oltextbuff.h>
...
int OlNumCharsInTextBufferLine(OlTextBufferPtr text,
  TextLine line);
```

Arguments

`text`	The text buffer
`line`	The line to get the number of chars for

Description

The function `OlNumCharsInTextBufferLine()` returns the number of characters in the specified line.

OlNumUnitsInTextBufferLine()

Synopsis

```
#include <Oltextbuff.h>
...
int OlNumUnitsInTextBufferLine(OlTextBufferPtr text,
  TextLine line);
```

Arguments

`text`	The text buffer
`line`	The line to get the number of units for

Description

The function `OlNumCharsInTextBufferLine()` returns the number of units in the specified line.

OlPositionOfLine()

Synopsis

```
#include <Oltextbuff.h>
...
TextPosition OlPositionOfLine(OlTextBufferPtr text,
  TextLine lineindex);
```

Arguments

`text`	The text buffer
`lineindex`	The line index to translate into a text position

Description

The `OlPositionOfLine()` function is used to translate a `lineindex` in the `OlTextBuffer` to a `TextPosition`. It returns the translated `TextPosition` or EOF if the `lineindex` is invalid.

OlPositionOfLocation()

Synopsis

```
#include <Oltextbuff.h>
...
TextPosition OlPositionOfLocation(OlTextBufferPtr text,
  TextLocation* location);
```

Arguments

`text`	The text buffer
`location`	The location to translate to a text position

Description

The `OlPositionOfLocation()` function is used to translate a location in the `OlTextBuffer` to a `TextPosition`. It returns the translated `TextPosition` or `EOF` if the location is invalid.

OlPreviousLocation()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlPreviousLocation(OlTextBufferPtr text,
  TextLocation* current);
```

Arguments

`text`	The text buffer
`current`	The current location

Description

The `OlPreviousLocation()` function returns the pointer to the `TextLocation` which precedes the given current location in a `OlTextBuffer`. If the current location points to the beginning of the `OlTextBuffer`, this function wraps.

Note – The current location is modified. It contains the previous location at the end of the call.

See Also

`OlNextLocation()`

OlPreviousTextBufferWord()

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlPreviousTextBufferWord(OlTextBufferPtr text,
  TextLocation* current);
```

Arguments

`text`	The text buffer
`current`	The current location

Description

The `OlPreviousTextBufferWord()` function is used to locate the beginning of a word in a `OlTextBuffer` relative to a given current location. It returns the location of the beginning of the word which precedes the given current location. If the current location is within a word, this function will skip over the current word. If the current word is the first word in the `OlTextBuffer`, the function wraps to the end of the buffer.

Note – The location passed to this function is modified. It contains the start of the previous buffer word at the end of the call.

See Also

`OlPreviousTextBufferWord()`

OlReadFileIntoTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
OlTextBufferPtr OlReadFileIntoTextBuffer(OlStrRep strrep,
  char* filename,
  TextUpdateFunction update_func,
  XtPointer data);
```

Arguments

text	The text buffer
update_func	The update function
data	Data

Description

The `OlReadFileIntoTextBuffer()` function is used to read the given file into a newly allocated `OlTextBuffer`. The supplied `TextUpdateFunction` and data pointer are associated with this `OlTextBuffer`.

See Also

`OlReadStringIntoTextBuffer()`

OlReadIntoTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
```

Arguments

text	The text buffer

Description

The `OlReadStringIntoTextBuffer()` function is used to copy the given string into a newly allocated `OlTextBuffer`. The supplied `TextUpdateFunction` and data pointer are associated with this `OlTextBuffer`.

See Also

`OlReadFileIntoTextBuffer()`

OlRegisterAllTextBufferScanFunctions()

Synopsis

```
#include <Oltextbuff.h>
...
void OlRegisterAllTextBufferScanFunctions(OlStrRep strrep,
  OlStrScanDefFunc forward_scan_func,
  OlStrScanDefFunc backward_scan_func);
```

Arguments

`strrep`	The string representation (`SB_STR_REP`, `WC_STR_REP`, or `MB_STR_REP`
`forward_scan_func`	The forward scan function to be used by all MLTextBuffers
`backward_scan_func`	The backward scan function to be used by all MLTextBuffers

The functions `forward_scan_func` and `backward_scan_func` specify `OlStrScanDefFunc` functions. `OlStrScanDefFunc` is defined as:

```
typedef XtPointer      (*OlStrScanDefFunc)(OlStr string,
                 OlStr curp,
                 OlStr expression);
```

Description

The `OlRegisterAllTextBufferWordDefinition()` procedure provides the capability to replace the text buffer functions used by all `OlTextBuffers`.

OlRegisterAllTextBufferWordDefinition()

Synopsis

```
#include <Oltextbuff.h>
...
void OlRegisterAllTextBufferWordDefinition(OlStrRep strrep,
  OlStrWordDefFunc word_definition_func);
```

Arguments

strrep	The string representation
word_definition_func	The word definition function used by all MLTextBuffers

word_definition_func specifies a OlStrWordDefFunc(), which is defined as:

```
typedef Boolean        (*OlStrWordDefFunc)(OlStr rc);
```

Description

The OlRegisterAllTextBufferWordDefinition() procedure provides the capability to replace the word definition function used by all OlTextBuffers. These functions are responsible for returning True if the character c points to is considered a character that can occur in a word and False otherwise. Calling this function with NULL reinstates the default word definition function associated with the text format.

OlRegisterPerTextBufferScanFunctions()

Synopsis

```
#include <Oltextbuff.h>
...
void OlRegisterPerTextBufferScanFunctions(OlTextBufferPtr text,
  OlStrScanDefFunc forward_scan_func,
  OlStrScanDefFunc backward_scan_func);
```

Arguments

strrep	The string representation (OL_SB_STR_REP, OL_WC_STR_REP, or OL_MB_STR_REP)
forward_scan_func	The forward scan function used by MLTextBuffers
backward_scan_func	The backward scan function used by MLTextBuffers

forward_scan_func and backward_scan_func specify OlStrScanDefFunc functions. OlStrScanDefFunc is defined as:

```
typedef XtPointer       (*OlStrScanDefFunc)(OlStr string,
                 OlStr curp,
                 OlStr expression);
```

Description

The `OlRegisterPerTextBufferScanFunctions()` procedure provides the capability to replace the scan functions used by the `OlForwardScanTextBuffer()` and `OlBackwardScanTextBuffer()` functions, as applied to the passed `OlTextBuffer` only.

The `OlRegisterAllTextBufferScanFunctions()` procedure provides the capability to replace the scan functions used by the `OlForwardScanTextBuffer` and `OlBackwardScanTextBuffer` functions, as applied to all `OlTextBuffers`. These functions are responsible for returning either a pointer to the beginning of a match for the expression or `NULL`. Calling this procedure with NULL function pointers reinstates the default regular expression facility associated with the text format.

OlRegisterPerTextBufferWordDefinition()

Synopsis

```
#include <Oltextbuff.h>
...
void OlRegisterPerTextBufferWordDefinition(OlTextBufferPtr text,
  OlStrWordDefFunc word_definition_func);
```

Arguments

`text`	The text buffer
`word_definition_func`	The new word definition function

`word_definition_func` specifies a `OlStrWordDefFunc` which is defined as:

```
typedef Boolean         (*OlStrWordDefFunc)(OlStr rc);
```

Description

The `OlRegisterPerTextBufferWordDefinition()` procedure provides the capability to replace the word definition function used by the passed `OlTextBuffer`.

OlRegisterTextBufferUpdate()

Synopsis

```
#include <Oltextbuff.h>
...
void OlRegisterTextBufferUpdate(OlTextBufferPtr text,
  TextUpdateFunction update_func,
  XtPointer data);
```

Arguments

`text`	The text buffer
`update_func`	The update function
`data`	Data

Description

The `OlRegisterTextBufferUpdate()` procedure associates the `TextUpdateFunction`, `update_func` and data pointer `data` with the given `OlTextBuffer` text. This update function is called whenever an update operation is performed on the `OlTextBuffer`. (See `OlReplaceBlockInTextBuffer()` for more details.)

Note – Calling this function increments a reference count mechanism used to determine when to actually free the `OlTextBuffer`. Calling the function with a NULL value for the function circumvents this mechanism.

See Also

`OlUnregisterTextBufferUpdate()`, `OlReadStringIntoTextBuffer()`, `OlReadFileIntoTextBuffer()`

OlReplaceBlockInTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
EditResult OlReplaceBlockInTextBuffer(OlTextBufferPtr text,
  TextLocation* startloc,
  TextLocation* endloc,
  OlStr string,
  TextUpdateFunction update_func,
  XtPointer data);
```

Arguments

`text`	The text buffer
`startloc`	The start location
`endloc`	the end location
`string`	The string to replace the block with
`update_func`	The update function
`data`	Data

Description

The `OlReplaceBlockInTextBuffer()` function is used to update the contents of the `OlTextBuffer` associated with text. The characters stored between `startloc` (inclusive) and `endloc` (exclusive) are deleted and the string is inserted after `startloc`. If the edit succeeds the `TextUpdateFunction f` is called with the parameters: `d`, `text`, and `1`; then any other different update functions associated with the `OlTextBuffer` are called with their associated data pointer, `text`, and `0`. This function records the operation performed in `OlTextUndoItem` structures. The contents of these structures can be used to implement an undo function. The contents can also be used to determine the type of operation performed. A structure is allocated for both the delete and insert information. The hints provided in these structures is the inclusive OR of

```
TEXT_BUFFER_NOP
TEXT_BUFFER_INSERT_LINE
TEXT_BUFFER_INSERT_SPLIT_LINE
TEXT_BUFFER_INSERT_CHARS
TEXT_BUFFER_DELETE_END_LINE
```

```
TEXT_BUFFER_DELETE_START_CHARS
TEXT_BUFFER_DELETE_END_CHARS
TEXT_BUFFER_DELETE_JOIN_LINE
TEXT_BUFFER_DELETE_SIMPLE
```

See Also

`OlReplaceCharInTextBuffer()`

OlReplaceCharInTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
EditResult OlReplaceCharInTextBuffer(OlTextBufferPtr text,
  TextLocation* location,
  OlStr c,
  TextUpdateFunction update_func,
  XtPointer data);
```

Description

The `OlReplaceCharInTextBuffer()` function is used to replace the character in the `OlTextBuffer` text at `location` with the character `c`.

See Also

`OlReplaceBlockInTextBuffer()`

OlSaveTextBuffer()

Synopsis

```
#include <Oltextbuff.h>
...
SaveResult OlSaveTextBuffer(OlTextBufferPtr text,
  char* filename);
```

Arguments

`text`	The text buffer
`filename`	The filename to write the text buffer to

Description

The `OlSaveTextBuffer()` function is used to write the contents of the `OlTextBuffer` to the file, `filename`. It returns a `SaveResult` which can be either `SAVE_FAILURE` or `SAVE_SUCCESS`.

OlSetTextUndoDeleteItem()

Synopsis

```
#include <Oltextbuff.h>
...
void OlSetTextUndoDeleteItem(OlTextBufferPtr text,
  OlTextUndoItem text_undo_deleted);
```

Arguments

`text`	The text buffer
`text_undo_deleted`	The item for which the delete was undone

Description

The function, `OlSetTextUndoDeleteItem()` sets the "deleted" `OlTextUndoItem` of the `OlTextBuffer` to the value of the passed `OlTextUndoItem`. The "deleted" string is copied in. `OlTextUndoItem` is defined as:

```
typedef struct _OlTextUndoItem {
  OlStr string;
  TextLocation start;
  TextLocation end;
  TextUndoHint hint;
} OlTextUndoItem;
```

OlSetTextUndoInsertItem()

Synopsis

```
#include <Oltextbuff.h>
...
void OlSetTextUndoInsertItem(OlTextBufferPtr text,
  OlTextUndoItem text_undo_insert);
```

Arguments

`text`	The text buffer
`lineindex`	The item for which the insert was undone

Description

The function, `OlSetTextUndoInsertItem()`, sets the "insert" `OlTextUndoItem` of the `OlTextBuffer` to the value of the passed `OlTextUndoItem`. The "insert" string is copied in `OlStartCurrentTextBufferWord()`.

Synopsis

```
#include <Oltextbuff.h>
...
TextLocation* OlStartCurrentTextBufferWord(OlTextBufferPtr text,
  TextLocation* current);
```

Arguments

`text`	The text buffer
`current`	The location

Description

The `OlStartCurrentTextBufferWord()` function is used to locate the beginning of a word in the `OlTextBuffer` relative to a given current location. It returns the location of the beginning of the current word.

Note – This return value will equal the given current value if the current location is the beginning of a word. If the location is not in a word, it returns the start of the "*not word*" region it is in.

Warning – The location passed to this function is modified. It contains the start of the current buffer word (or "*not word*") at the end of the call.

See Also

`MPreviousTextBufferWord(), OlNextTextBufferWord()`

OlUnitOffsetOfLocation()

Synopsis

```
#include <Oltextbuff.h>
...
UnitPosition OlUnitOffsetOfLocation(OlTextBufferPtr text,
  TextLocation* loc);
```

Arguments

`text`	The text buffer
`loc`	The location

Description

The function, `OlUnitOffsetOfLocation()`, returns the unit offset corresponding to the `TextLocation` offset passed to it. The units are `char` for single-byte and multi-byte and `w_char` for wide character.

OlUnregisterTextBufferUpdate()

Synopsis

```
#include <Oltextbuff.h>
...
int OlUnregisterTextBufferUpdate(OlTextBufferPtr text,
  TextUpdateFunction update_func,
  XtPointer data);
```

Arguments

`text`	The text buffer
`update_func`	The update function to disassociate
`data`	Data

Description

The `OlUnregisterTextBufferUpdate()` function disassociates the `TextUpdateFunction` `f` and data pointer data with the given `OlTextBuffer` text. If the function/data pointer pair is not associated with the given `OlTextBuffer`, `0` is returned; otherwise the association is dissolved and `1` is returned.

See Also

`OlRegisterTextBufferUpdate()`, `OlFreeTextBuffer()`

FileChooser Example

The following is a sample program showing an implementation of a text editor that uses `FileChooser` to interface with the filesystem.

Code Example 16-9 texteditor.c

```
/*
 *  Copyright (C) 1992, 1993 Sun Microsystems, Inc
 *          All rights reserved.
 *      Notice of copyright on this source code
 *      product does not indicate publication.
 *
 * RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by
```

Code Example 16-9 texteditor.c (Continued)

```
 * the U.S. Government is subject to restrictions as set forth
 * in subparagraph (c)(1)(ii) of the Rights in Technical Data
 * and Computer Software Clause at DFARS 252.227-7013 (Oct. 1988)
 * and FAR 52.227-19 (c) (June 1987).
 *
 * This file is a product of Sun Microsystems, Inc. and is
 * provided for unrestricted use provided that this legend is
 * included on all media and as a part of the software
 * program in whole or part. Users may copy or modify this
 * file without charge, but are not authorized to license or
 * distribute it to anyone else except as part of a product or
 * program developed by the user.
 *
 * THIS FILE IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND
 * INCLUDING THE WARRANTIES OF DESIGN, MERCHANTIBILITY AND
 * FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE
 * OF DEALING, USAGE OR TRADE PRACTICE.
 *
 * This file is provided with no support and without any
 * obligation on the part of Sun Microsystems, Inc. to assist
 * in its use, correction, modification or enhancement.
 *
 * SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT
 * TO THE INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY
 * PATENTS BY THIS FILE OR ANY PART THEREOF.
 *
 * In no event will Sun Microsystems, Inc. be liable for any
 * lost revenue or profits or other special, indirect and
 * consequential damages, even if Sun has been advised of the
 * possibility of such damages.
 *
 *  Sun Microsystems, Inc., 2550 Garcia Avenue,
 *  Mountain View, California 94043.
 *
 */


/**************************************************************************
 *
 *  Implementation of a simple OLIT text editor
 *
 **************************************************************************/


/**************************************************************************
```

Code Example 16-9 texteditor.c (Continued)

```
 *
 * Imported Interfaces
 *
 ***********************************************************************/

#include <errno.h> /* errno */
#include <libintl.h>/* dgettext() */
#include <libgen.h>/* basename() */
#include <limits.h>/* MAX_PATH */

#ifndefMAX_PATH
    #include <sys/param.h>/* MAXPATHLEN */
#endif/* MAX_PATH */

#include <stdio.h>/* [sf]printf() */
#include <stdlib.h>/* exit(), EXIT_FAILURE */
#include <string.h>
#include <widec.h>

#include <X11/IntrinsicP.h>
#include <X11/StringDefs.h>

#include <Xol/FileCh.h>
#include <Xol/FileChSh.h>
#include <Xol/Form.h>
#include <Xol/MenuButton.h>
#include <Xol/OblongButt.h>
#include <Xol/OpenLook.h>
#include <Xol/ScrolledWi.h>
#include <Xol/TextEdit.h>


/***********************************************************************
 *
 * Module Private Macro Definitions
 *
 ***********************************************************************/

#ifdefMAX_PATH
    #defineBIGGEST_PATH(MAX_PATH + 1)
#else
    #defineBIGGEST_PATH(MAXPATHLEN)
#endif/* MAX_PATH */

#defineMessagesDomain"oltextedit"
```

Code Example 16-9 texteditor.c (Continued)

```
#ifdefDEBUG
#defineFPRINTFfprintf
#else
#defineFPRINTF(void)
#endif/* DEBUG */


/***********************************************************************
 *
 * Public Private Storage Allocation
 *
 ***********************************************************************/

static charlc_messages[2 * BUFSIZ];
static Widgetbasewin;/* top level widget */
static Widgetopen_ob;

static const char* fallback_resources[] = {
        "*Background:gray",
        "*ApplicationShell.title:OLIT Text Editor (Demo)",

        "*file_mb.label:File",

        "*new_ob.label:New",
        "*open_ob.label:Open...",
        "*save_ob.label:Save",
        "*save_as_ob.label:Save As...",
        "*include_ob.label:Include...",
        "*print_one_ob.label:Print One",
        "*print_ob.label:Print...",

        "*text_te.font:\
            -*-lucida*-medium-r-normal-sans-*-120-*-*-*-*-*-*",

        "*text_te.borderWidth:1",
        "*text_te.height:610",
        "*text_te.width:595",

        /* "*text_te.imPreeditStyle:overTheSpot", */

        "*text_te.source:\
This is a demonstration program of the OLIT file chooser facility. It is a very
simple OPEN LOOK text editor. The text that you are reading is in a TextEdit
widget and can be edited.",
```

Code Example 16-9 texteditor.c (Continued)

```
        NULL
};


/***********************************************************************
 *
 * Forward Declaration Of Module Private Functions
 *
 ***********************************************************************/

static voidconstruct_basewin(const Widget parent);

static voidupdate_footers(const String left_string,
    const String right_string);

static voidupdate_left_footer(const String string);
static voidupdate_right_footer(const String string);

static voidtext_changed_cb(Widget widget, XtPointer client_data,
           XtPointer call_data);

static voidselect_cb(Widget widget, XtPointer client_data,
           XtPointer call_data);

static voidnew_cb(Widget widget, XtPointer client_data,
    XtPointer call_data);

static voidsave_cb(Widget widget, XtPointer client_data,
           XtPointer call_data);

static voidio_cb(Widget widget, XtPointer client_data,
           XtPointer call_data);

static voidprint_cb(Widget widget, XtPointer client_data,
           XtPointer call_data);

static voidsave_document(const TextEditWidget tew, const String path_name,
    const Widget open_ob);

static voidinsert_document(const TextEditWidget tew, const String path_name);


/***********************************************************************
 *
```

Code Example 16-9 texteditor.c (Continued)

```
 * Implementation Of this Module's External Functions
 *
 ***********************************************************************/


/***********************************************************************
 *
 * main --
 *
 ***********************************************************************/

void
main(const int argc, const char*const argv[])
{
    XtAppContextapp_context;
    String     openwin_path;
    char domain_path[BIGGEST_PATH];
    char messages_file[BIGGEST_PATH];
    String     locale;
    String     lang_locale;

    /* Make sure the syscall error handler is reset */
 errno = 0;


    /* Initialize I18N environment */

 if (NULL == (openwin_path = getenv("OPENWINHOME"))) {
 (void) sprintf(lc_messages, "%s", dgettext(MessagesDomain,
    "Please set OPENWINHOME"));
 (void) fprintf(stderr, "%s", lc_messages);
 exit(EXIT_FAILURE);
 }

 locale = (NULL == (lang_locale = getenv("LANG"))) ? "C" : lang_locale;

    (void) sprintf(messages_file, "%s/lib/locale/%s/LC_MESSAGES/%s.mo",
        openwin_path, locale, MessagesDomain);

    (void) sprintf(domain_path, "%s/lib/locale", openwin_path);

    if (NULL != fopen(messages_file, "r"))
 (void) bindtextdomain(MessagesDomain, domain_path);
    else
        (void) bindtextdomain(MessagesDomain, getenv("PWD"));
```

Code Example 16-9 texteditor.c (Continued)

```
    /* Initialize OLIT and the Intrinsics */

    (void) OlToolkitInitialize(NULL);

    OlSetDefaultTextFormat(OL_MB_STR_REP);

    basewin = XtVaAppInitialize(&app_context, (String)NULL,
        (XrmOptionDescList)NULL, 0, (int*)&argc, (String*)argv,
        (String*)fallback_resources,
            XtNfooterPresent,TRUE,
        NULL);

    construct_basewin(basewin);

    XtRealizeWidget(basewin);

    XtAppMainLoop(app_context);
    /*NOTREACHED*/
    exit(EXIT_FAILURE);
    /*NOTREACHED*/
} /* end of main() */


/**************************************************************************
 *
 * Implementation Of this Module's Internal Functions
 *
 **************************************************************************/


/**************************************************************************
 *
 * construct_basewin -- Populate the top level panel
 *
 **************************************************************************/

static void
construct_basewin(const Widget parent)
{
    #define MAX_CHILDREN(8)
    Widget     children[MAX_CHILDREN];
    Cardinalnum_children;
```

Code Example 16-9 texteditor.c (Continued)

```
    Widget panel_f,file_mb,file_m,new_ob,open_fcs,open_fc,save_ob,save_fcs,
           save_fc,save_as_ob,save_as_fcs,save_as_fc,include_ob,include_fcs,
           include_fc,spacer_ob,print_one_ob,print_ob,text_sw,text_te;

    /* Instantiate the widget tree */

    panel_f = XtVaCreateWidget("panel_f", formWidgetClass, parent, NULL);

    file_mb = XtVaCreateWidget("file_mb", menuButtonWidgetClass,
        panel_f, NULL);

    XtVaGetValues(file_mb, XtNmenuPane, &file_m, NULL);

    /*
     * Text editing pane
     */
    text_sw = XtVaCreateWidget("text_sw", scrolledWindowWidgetClass, panel_f,
           XtNyAddHeight,TRUE,
           XtNyRefWidget,file_mb,
           XtNxAttachRight,TRUE,
           XtNyAttachBottom,TRUE,
           XtNxResizable,TRUE,
           XtNyResizable,TRUE,
        NULL);

    text_te = XtVaCreateWidget("text_te", textEditWidgetClass, text_sw,
           XtNtextFormat,OL_WC_STR_REP,
           XtNuserData, NULL,
        NULL);

    XtVaSetValues(file_mb, XtNuserData, text_te, NULL);

    /*
     * New
     */
    new_ob = XtVaCreateWidget("new_ob", oblongButtonWidgetClass, file_m,
           XtNuserData,file_mb,
        NULL);

    /*
     * Open
     */
    open_ob = XtVaCreateWidget("open_ob", oblongButtonWidgetClass, file_m,
           XtNdefault,TRUE,
           XtNuserData,file_mb,
```

Code Example 16-9 texteditor.c (Continued)

```
        NULL);

    XtAddCallback(new_ob, XtNselect, new_cb, open_ob);

    open_fcs = XtVaCreatePopupShell("open_fcs", fileChooserShellWidgetClass,
        open_ob, NULL);

    XtVaGetValues(open_fcs, XtNfileChooserWidget, &open_fc, NULL);
    XtAddCallback(open_ob, XtNselect, select_cb, open_fcs);
    XtAddCallback(open_fc, XtNinputDocumentCallback, io_cb, text_te);

    /*
     * Save
     */
    save_ob = XtVaCreateWidget("save_ob", oblongButtonWidgetClass, file_m,
            XtNuserData,file_mb,
        NULL);

    XtAddCallback(text_te, XtNpostModifyNotification, text_changed_cb,
        save_ob);

    save_fcs = XtVaCreatePopupShell("save_fcs", fileChooserShellWidgetClass,
        save_ob,
            XtNoperation,OL_SAVE,
        NULL);

    XtVaGetValues(save_fcs, XtNfileChooserWidget, &save_fc, NULL);
    XtAddCallback(save_ob, XtNselect, save_cb, save_fcs);
    XtAddCallback(save_fc, XtNoutputDocumentCallback, io_cb, NULL);

    /*
     * Save As
     */
    save_as_ob = XtVaCreateWidget("save_as_ob", oblongButtonWidgetClass,
        file_m,
            XtNuserData,file_mb,
        NULL);

    save_as_fcs = XtVaCreatePopupShell("save_as_fcs",
        fileChooserShellWidgetClass, save_as_ob,
            XtNoperation, OL_SAVE_AS,
        NULL);

    XtVaGetValues(save_as_fcs, XtNfileChooserWidget, &save_as_fc, NULL);
    XtVaSetValues(save_fc, XtNuserData, save_as_fc, NULL);
```

Code Example 16-9 texteditor.c (Continued)

```
    XtVaSetValues(open_fc, XtNuserData, save_as_fc, NULL);
    XtAddCallback(save_as_ob, XtNselect, select_cb, save_as_fcs);
    XtAddCallback(save_as_fc, XtNoutputDocumentCallback, io_cb, NULL);

    /*
     * Include
     */
    include_ob = XtVaCreateWidget("include_ob", oblongButtonWidgetClass,
        file_m,
            XtNuserData,file_mb,
        NULL);

    include_fcs = XtVaCreatePopupShell("include_fcs",
        fileChooserShellWidgetClass, include_ob,
            XtNoperation, OL_INCLUDE,
        NULL);

    XtVaGetValues(include_fcs, XtNfileChooserWidget, &include_fc, NULL);
    XtAddCallback(include_ob, XtNselect, select_cb, include_fcs);
    XtAddCallback(include_fc, XtNinputDocumentCallback, io_cb, NULL);

    /*
     * Spacer
     */
    spacer_ob = XtVaCreateWidget("spacer_ob", oblongButtonWidgetClass,
        file_m,
            XtNsensitive,FALSE,
            XtNlabel,"",
        NULL);

    /*
     * Print One
     */
    print_one_ob = XtVaCreateWidget("print_one_ob", oblongButtonWidgetClass,
        file_m,
            XtNuserData,file_mb,
        NULL);

    XtAddCallback(print_one_ob, XtNselect, print_cb, NULL);

    /*
     * Print
     */
    print_ob = XtVaCreateWidget("print_ob", oblongButtonWidgetClass,
        file_m,
```

Code Example 16-9 texteditor.c (Continued)

```
            XtNuserData,file_mb,
        NULL);

    XtAddCallback(print_ob, XtNselect, print_cb, NULL);


    /* Manage components by their respective parents */
    num_children = 0;
    children[num_children++] = new_ob;
    children[num_children++] = open_ob;
    children[num_children++] = save_ob;
    children[num_children++] = save_as_ob;
    children[num_children++] = include_ob;
    children[num_children++] = spacer_ob;
    children[num_children++] = print_one_ob;
    children[num_children++] = print_ob;
    XtManageChildren(children, num_children);

    XtManageChild(text_te);

    num_children = 0;
    children[num_children++] = file_mb;
    children[num_children++] = text_sw;
    XtManageChildren(children, num_children);

    XtManageChild(panel_f);

    #undef MAX_CHILDREN

} /* end of construct_basewin() */


/***************************************************************************
 *
 * update_footers --
 *
 ***************************************************************************/

static void
update_footers(const String left_string, const String right_string)
{

    update_left_footer(left_string);
    update_right_footer(right_string);
} /* end of update_footers() */
```

Code Example 16-9 texteditor.c (Continued)

```
/**********************************************************************
 *
 * update_left_footer --
 *
 **********************************************************************/

static void
update_left_footer(const String string)
{

    if (NULL != string)
        XtVaSetValues(basewin, XtNleftFooterString, string, NULL);
} /* end of update_left_footer() */


/**********************************************************************
 *
 * update_right_footer --
 *
 **********************************************************************/

static void
update_right_footer(const String string)
{

    if (NULL != string)
        XtVaSetValues(basewin, XtNrightFooterString, string, NULL);
} /* end of update_right_footer() */


/**********************************************************************
 *
 * text_changed_cb --
 *
 **********************************************************************/

/*ARGSUSED*/
static void
text_changed_cb(
    Widget     widget, /* unused */
    XtPointerclient_data,
    XtPointercall_data/* unused */
)
```

Code Example 16-9 texteditor.c (Continued)

```
{

    XtVaSetValues((Widget)client_data, XtNdefault, TRUE, NULL);

} /* end of text_changed_cb() */


/************************************************************************
 *
 * new_cb --
 *
 ************************************************************************/

/*ARGSUSED2*/
static void
new_cb(
    Widget      widget,
    XtPointerclient_data,
    XtPointercall_data/* unused */
)
{
    Widget      text_te;
    Widget      file_mb;
    String      old_path;

    XtVaGetValues(widget, XtNuserData, &file_mb, NULL);
    XtVaGetValues(file_mb, XtNuserData, &text_te, NULL);

    XtVaGetValues(text_te, XtNuserData, &old_path, NULL);
    XtFree(old_path);

    XtVaSetValues(text_te,
            XtNsource,"",
            XtNuserData,NULL,
        NULL);

    update_footers("", "");

    XtVaSetValues((Widget)client_data, XtNdefault, TRUE, NULL);
} /* end of new_cb() */


/************************************************************************
 *
 * select_cb --
```

Code Example 16-9 texteditor.c (Continued)

```
 *
 ***********************************************************************/

/*ARGSUSED*/
static void
select_cb(
    Widget     widget, /* unused */
    XtPointerclient_data,
    XtPointercall_data/* unused */
)
{
    const Widgetshellw = (Widget)client_data;

    XtPopup(shellw, XtGrabNone);
    /* if already popped up than raise it to top */

    update_footers("", "");
} /* end of select_cb() */


/***********************************************************************
 *
 * save_cb --
 *
 ***********************************************************************/

/*ARGSUSED2*/
static void
save_cb(
    Widget     widget,
    XtPointerclient_data,
    XtPointercall_data/* unused */
)
{
    Widget     text_te;
    Widget     file_mb;
    String     path_name;

    XtVaGetValues(widget, XtNuserData, &file_mb, NULL);
    XtVaGetValues(file_mb, XtNuserData, &text_te, NULL);
    XtVaGetValues(text_te, XtNuserData, &path_name, NULL);

    if (NULL == path_name || '\0' == *path_name) {
        XtPopup((Widget)client_data, XtGrabNone);
        update_footers("", "");
```

Code Example 16-9 texteditor.c (Continued)

```
    } else
        save_document((TextEditWidget)text_te, path_name, open_ob);
} /* end of save_cb() */


/*********************************************************************
 *
 * io_cb --
 *
 *********************************************************************/

/*ARGSUSED1*/
static void
io_cb(
    Widget     widget,
    XtPointerclient_data,/* unused */
    XtPointercall_data
)
{
    static charpath_buf[BIGGEST_PATH] = { '\0' };

    Widget     file_mb;
    Widget     text_te;
    Widget     save_as_fc;
    OlAnyCallbackStruct*any_cd = call_data;

    /* file chooser ... file chooser popup ... file menu item */
    XtVaGetValues(XtParent(XtParent(widget)),
            XtNuserData,&file_mb,
        NULL);

    XtVaGetValues(file_mb, XtNuserData, &text_te, NULL);

    switch (any_cd->reason) {
    case OL_REASON_INPUT_DOCUMENT: {
        OlFileChDocumentCallbackStruct* cd = call_data;

        (void) sprintf(path_buf, "%s/%s", cd->request_document_folder,
            cd->request_document);

        switch (cd->operation) {
        case OL_OPEN:
            XtVaSetValues(text_te,
                    XtNwrapMode,OL_WRAP_OFF,
                    XtNsourceType,OL_DISK_SOURCE,
```

Code Example 16-9 texteditor.c (Continued)

```
                XtNsource,path_buf,
                XtNuserData,XtNewString(path_buf),
            NULL);

         (void) sprintf(lc_messages, dgettext(MessagesDomain,
             "Opened document %s"), path_buf);
         update_left_footer(lc_messages);

         XtVaGetValues(widget, XtNuserData, &save_as_fc, NULL);

         XtVaSetValues(save_as_fc,
                XtNlastDocumentName, basename(path_buf),
            NULL);

         break;

     case OL_INCLUDE:
         insert_document((TextEditWidget)text_te, path_buf);
         break;
     }

     }
     break;
 case OL_REASON_OUTPUT_DOCUMENT: {
     OlFileChDocumentCallbackStruct*cd = call_data;

     (void) sprintf(path_buf, "%s/%s", cd->request_document_folder,
         cd->request_document);

     save_document((TextEditWidget)text_te, path_buf, open_ob);

     if (OL_SAVE == cd->operation) {
         XtVaSetValues(text_te,
                XtNuserData,XtNewString(path_buf),
            NULL);
         XtVaGetValues(widget, XtNuserData, &save_as_fc, NULL);

         XtVaSetValues(save_as_fc,
                XtNlastDocumentName, basename(path_buf),
            NULL);
     }

     }
     break;
 }
```

Code Example 16-9 texteditor.c (Continued)

```
} /* end of io_cb() */


/*********************************************************************
 *
 * print_cb --
 *
 *********************************************************************/

/*ARGSUSED0*/
static void
print_cb(
    Widget     widget, /* unused */
    XtPointerclient_data,/* unused */
    XtPointercall_data/* unused */
)
{

    (void) sprintf(lc_messages, dgettext(MessagesDomain,
        "Printing is not implemented in this application."));
    update_footers(lc_messages, "");
} /* end of print_cb() */


/*********************************************************************
 *
 * save_document --
 *
 *********************************************************************/

static void
save_document(const TextEditWidget tew, const String path_name,
    const Widget open_ob)
{
    OlStrReptext_format;
    SaveResultsave_result;

    XtVaGetValues((Widget)tew, XtNtextFormat, &text_format, NULL);

    switch (text_format) {
    case OL_SB_STR_REP:
        save_result = SaveTextBuffer(OlTextEditTextBuffer(tew),
            path_name);
        break;
    case OL_MB_STR_REP:
```

Code Example 16-9 texteditor.c (Continued)

```
    /*FALLTHROUGH*/
    case OL_WC_STR_REP:
        save_result = OlSaveTextBuffer(OlTextEditOlTextBuffer(tew),
            path_name);
        break;
    }

    (void) sprintf(lc_messages, dgettext(MessagesDomain,
        SAVE_FAILURE == save_result ? "Save to document %s failed!" :
            "Saved to document %s"), path_name);
    update_footers(lc_messages, "");

    if (SAVE_SUCCESS == save_result)
        XtVaSetValues(open_ob, XtNdefault, TRUE, NULL);
} /* end of save_document() */


/**********************************************************************
 *
 * insert_document --
 *
 **********************************************************************/

static void
insert_document(const TextEditWidget tew, const String path_name)
{
    #define WC_BUFSIZ(BUFSIZ / 4)

    OlStrReptext_format;
    FILE*   fp;
    char    buf[BUFSIZ];
    wchar_t wc_buf[WC_BUFSIZ];

    if (NULL == (fp = fopen(path_name, "r"))) {
        (void) sprintf(lc_messages, dgettext(MessagesDomain,
            "Could not open document %s for insertion!"), path_name);
        update_footers(lc_messages, "");
        return;
    }

    XtVaGetValues((Widget)tew, XtNtextFormat, &text_format, NULL);

    (void) OlTextEditUpdate(tew, FALSE);

    while (NULL != fgets(buf, sizeof buf, fp)) {
```

Code Example 16-9 texteditor.c (Continued)

```
        if (OL_WC_STR_REP == text_format)
            (void) mbstowcs(wc_buf, buf, WC_BUFSIZ);

        if (EDIT_SUCCESS != OlTextEditInsert(tew,
                (OL_WC_STR_REP == text_format) ?
                    (char*)wc_buf : buf,
                BUFSIZ)) {

            (void) sprintf(lc_messages,
                dgettext(MessagesDomain,
                "Insertion of document %s failed!"),
                path_name);

            update_footers(lc_messages, "");
            return;
        }
    }

    (void) OlTextEditUpdate(tew, TRUE);

    (void) sprintf(lc_messages, dgettext(MessagesDomain,
        "Inserted document %s"), path_name);
    update_footers(lc_messages, "");
} /* end of insert_document() */


/* end of oltextedit.c */
```

OLIT Man Pages

TextLine Widget

Synopsis

```
#include <Xol/OpenLook.h>
#include <Xol/TextLine.h>
```

Class

Name: `TextLine`

Pointer: `textLineWidgetClass`

Ancestry

Core -> Primitive -> TextLine

Description

The `TextLine` widget is a one-line input field for text data. Once the input-focus is moved into the widget, keyboard entry is allowed. If the input value exceeds the length of the input field, the ScrollButtons appear. Hidden text can then be scrolled into view, by pressing the ScrollButtons. Pressing the buttons continuously, scrolls the text repeatedly with a user-adjustable delay.

The `TextLine` contains the following graphical elements:

- Right justified bold label at the left of the TextLine
- Input field
- Input Caret (not present in ReadOnly mode)
- 1 point (for Mono) or chiseled underline (not present in ReadOnly mode)
- Optional ScrollButtons

Keyboard Traversal

The `TextLine` allows keyboard entry if it is sensitive and it has the keyboard focus. However in ReadOnly mode, the widget is not traversable and it does not receive keyboard focus. Selection is also turned off during ReadOnly mode.

The `TextLine` responds to the following keyboard navigation keys:

- `NEXT_FIELD` moves to the next traversable widget in the window
- `PREV_FIELD` moves to the previous traversable widget in the window
- `CHARFWD` moves the caret forward one character
- `CHARBAK` moves the caret backward one character
- `WORDFWD` moves the caret forward one word
- `WORDBAK` moves the caret back one word
- `LINESTART` moves the caret to the beginning of the display
- `LINEEND` moves the caret to the end of the display
- `MENUKEY` posts the menu associated with the TextLine

The `TextLine` responds to the following edit keys:

- `DELCHARFWD` deletes the character to the right of the caret
- `DELCHARBAK` deletes the character to the left of the caret
- `DELWORDFWD` deletes the word to the right of the caret
- `DELWORDBAK` deletes the word to the left of the caret
- `DELLINEFWD` deletes to the end of the line from the caret
- `DELLINEBAK` deletes from the beginning of the line to the caret

- `DELLINE` deletes the line containing the caret
- `UNDO` undoes the last edit

Display of the Keyboard Mnemonic

The `TextLine` does not display any mnemonic. If the `TextLine` is the child of a `Caption` widget, the `Caption` can be used to display any mnemonic.

Display of the Keyboard Accelerator

The `TextLine` does not respond to any keyboard accelerators.

Display of Text

The `TextLine` displays its contents in the font specified by the `XtNfont` resource. If the length of the text exceeds the length of the input field, the widget sets up the Left or Right or both ScrollButtons to indicate this. The text is then visually truncated at the boundaries to show only as many characters as can fit in the input field. The truncation is always at a character boundary. A ScrollButton is present only if characters are hidden in the direction indicated by that buttons.

The user can scroll to show the hidden parts of the text by clicking or pressing the ScrollButtons. Clicking SELECT on any ScrollButton will scroll the text one character in the direction indicated by that button. Pressing SELECT on any ScrollButton will repeat the scrolling with a user-adjustable delay between each scroll.

Caret position

As characters are entered from the keyboard, the caret moves to the right until it reaches the right end of the input field. As additional characters are typed, the text jump-scrolls to the left by a specific amount. Note that the `TextLine` always keeps the cursor visible. Thus the presence or absence of either of the ScrollButtons is controlled by the current cursor position.

Selection of Text

Text selection can be done by using the mouse or keyboard. The widget also provides a set of convenience functions to manipulate the selection programmatically.

Coloration

See SnapShot of this feature in the OpenWindows chapter.

Core Resources

Name	Type	Default	Access
XtNx	Position	0	SGI
XtNy	Position	0	SGI
XtNwidth	Dimension	<Calculated>	SG
XtNheight	Dimension	<Calculated>	SG
XtNborderWidth	Dimension	0	SGI
XtNsensitive	Boolean	True	GIO
XtNancestorSensitive	Boolean	True	G
XtNscreen	Screen *	parent's	GI
XtNdepth	Cardinal	parent's	GI
XtNcolormap	Colormap	parent's	GI
XtNbackground	Pixel	parent's	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGI
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNmappedWhenMananaged	Boolean	True	SGI
XtNdestroyCallback	XtCallbackList	NULL	SGIO

Primitive Resources

Name	Type	Default	Access
XtNaccelerator	String	NULL	U
XtNacceleratorText	String	NULL	U
XtNconsumeEvent	Callback	NULL	SGIO
XtNfont	OlFont	XtDefaultFont	SGI
XtNfontColor	Pixel	XtDefaultForeground	SGI
XtNforeground	Pixel	XtDefaultForeground	SGI
XtNinputFocusColor	Pixel	"Red"	SGI
XtNmnemonic	OlRChar	NULL	SGI
XtNreferenceName	String	NULL	SGI
XtNreferenceWidget	OlStrRep	NULL	SGI
XtNtextFormat	OlStrRep	OL_SB_STR_REP	GI
XtNtraversalOn	Boolean	True	SGI
XtNuserData	XtPointer	NULL	SGI

TextLine Resources

- `XtNblinkRate`

Class:	`XtCBlinkRate`
Type:	`Int`
Def:	`1000`
Access:	SGI
Action:	Specifies the blink rate of the active cursor in terms of milliseconds. Value of zero turns blink off.
Values:	Any integer

- `XtNcaptionLabel`

Class:	`XtCCaptionLabel`
Type:	`OlStr`
Def:	`NULL`
Access:	SGI
Action:	Label for TextLine.
Values:	Any OlStr valid in the current locale

- `XtNcaptionFont`

Class:	`XtCCaptionFont`
Type:	`OlFont`
Def:	`OlDefaultBoldFont`
Access:	SI
Action:	Specifies the font for the caption label.
Values:	Any font valid in the current locale

- `XtNcharsVisible`

Class:	`XtCCharsVisible`
Type:	`Int`
Def:	`0`

Access:	GI
Action:	If non-zero, specifies the initial width of text in terms of characters. Overrides the `XtNwidth` setting. `XtNwidth` is then calculated as: `XtNwidth = charsVisible * max_char_width` over the given FontSet. Note that the actual number of characters visible could be more than the value of this resource - since `XtNwidth` is computed based on the max-char-width of the given FontSet. The resource value changes with Geometry-changes. Refer to the `XtNmaximumChars` resource below, too.
Values:	Any integer

- `XtNinitialDelay`

Class:	`XtCInitialDelay`
Type:	`Int`
Def:	`500`
Access:	SGI
Action:	Specifies the time in milliseconds of the initial repeat delay to be used when the scrolling arrows are pressed.
Values:	Any integer

- `XtNrepeatRate`

Class:	`XtCRepeatRate`
Type:	`Int`
Def:	`100`
Access:	SGI
Action:	Specifies the time in milliseconds of the repeat delay to be used when the scrolling arrows are pressed.
Values:	Any integer

- `XtNinsertTab`

Class:	`XtCInsertTab`
Type:	`Boolean`
Def:	`False`
Access:	SGI
Action:	Specifies whether <TAB>s are insertable.
Values:	True: <TAB> can be inserted. False: <TAB> acts as NEXTFIELD.

- `XtNmaximumChars`

Class:	`XtCMaximumChars`
Type:	`Int`
Def:	`0`
Access:	GI

Action:	Specifies the maximum internal buffer size in terms of characters. If zero, then the internal buffer will increase in size dynamically. If `XtNcharsVisible` is 0 (its default value), it is setup as described below:
	`if (charsVisible == 0)`
	`    charsVisible = (maximumChars ? maximumChars : 20)`

Values:	Any integer

- `XtNstring`

Class:	`XtCString`
Type:	`OlStr`
Def:	`NULL`
Access:	SGI
Action:	Contents of the TextLine buffer. On being set, the string is inserted into the widget's internal buffer. `XtGetValues` on this resource returns a pointer to the current data. Note that: (a) the widget should treat the pointed-to data as read-only. (b) the pointed-to data is guaranteed to be valid, only until the next Intrinsics call. Thus, if the application needs the data longer, it should make a copy of it.
Values:	Any `OlStr` valid in the current locale

- `XtNcursorPosition`

Class:	`XtCTextPosition`
Type:	`Int`
Def:	`0`
Access:	SGI
Action:	Specifies the position of the cursor. Setting this resource will cause scrolling if the new position is beyond the visual area.
Values:	0 <= `XtNcursorPosition` <= total number_of_characters

- XtNeditType

Class:	XtCEditType
Type:	OlDefine
Def:	OL_TEXT_EDIT
Access:	SGI
Action:	Controls the edit-mode.
Values:	OL_TEXT_EDIT / "text_edit" : The text is editable. OL_TEXT_READ / "text_read" : The text is ReadOnly. In ReadOnly mode, the input-caret is disabled and the widget is not traversable or selectable.

- XtNupdateDisplay

Class:	XtCUpdateDisplay
Type:	Boolean
Def:	True
Access:	SGI
Action:	Controls screen redisplay of screen. Useful during incremental programmatic edits.
Values:	False: Screen redisplay is stopped until it is set back to True. True: Redisplay the widget in the current state.

- XtNunderline

Type:	Boolean
Def:	True
Access:	SGI
Action:	Controls whether the underline is present or not.
Values:	True: Underline is present. False: Underline is absent. Note the OPEN LOOK specification states that the underline should be removed if the widget is in ReadOnly mode.

- XtNmenu

Class:	XtCReadOnly
Type:	Widget
Def:	NULL
Access:	GI
Action:	Provides a handle to the MenuShell widget which is popped up when you press the MENU key over the TextLine. By default, this menu contains the OPEN LOOK specified default elements:

	UNDO, CUT, COPY, PASTE, & DELETE. Programmers can augment this menu by creating more items as children of the MenuShell's `XtNmenuPane` widget, and they can also change attributes of the MenuShell or the default-items within. However, none of the default-items should be removed from the MenuShell. Programmers can also install their own Menu by setting this resource while creating the `TextLine` widget. In this case, the programmer's menu will be popped-up instead of the widget's built-in menu. In fact, the widget does not even create its own menu. Therefore, one can share menus among multiple `TextLine` widgets and thus avoid the overhead of having multiple MenuShells.
Values:	A valid widget ID.

- `XtNpreModifyCallback`

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGIO
Action:	This callback is invoked before modification of buffer is attempted.

The calldata structure is:

```
OlTLPreModifyCallbackStruct {
  int     reason,
  XEvent  *event,
  Booleanvalid,
  int     current_cursor,
  int     new_cursor,
  OlStrbuffer,
  int     start,
  int     replace_length,
  OlStrinsert_buffer,
  int     insert_length
}
```

Fields in the calldata structure are:

`reason`	`OL_REASON_PRE_MODIFICATION` or `OL_REASON_PROG_PRE_MODIFICATION`
`event`	corresponding `XEvent` structure
`valid`	If `True`, widget performs the modification, else doesn't.
`current_cursor`	Current position of cursor

`new_cursor`	Cursor position after modification.
`buffer`	A pointer to the `XtNstring` resource. Note that this pointer is ReadOnly and will be valid only within the callback.
`start`	Start of text to be deleted or replaced.
`replace_length`	Length of text to be deleted or replaced.
`insert_buffer`	The new string to be inserted.
`insert_length`	Length of insert-string.

All fields except "valid" are ReadOnly.

Note – `OL_REASON_PRE_MODIFICATION` indicates that the callback was invoked due to non-programmatic edits. `OL_REASON_PROG_PRE_MODIFICATION` indicates that this callback was invoked due to programmatic edits.

- XtNpostModifyCallback

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGIO
Action:	This callback is invoked after modification of buffer is done.

The calldata structure is:

```
OlTLPostModifyCallbackStruct {
 int   reason,
 XEvent  *event,
 int   cursor,
 OlStrbuffer,
}
```

Fields in the calldata structure are:

`reason`	`OL_REASON_POST_MODIFICATION` or
	`OL_REASON_PROG_POST_MODIFICATION`
`event`	corresponding `XEvent` structure
`cursor`	Current position of cursor.
`buffer`	A pointer to the `XtNstring` resource. Note that this pointer is ReadOnly and will be valid only within the callback.

All fields are ReadOnly.

Note – OL_REASON_POST_MODIFICATION indicates that the callback was invoked due to non-programmatic edits. OL_REASON_PROG_POST_MODIFICATION indicates that this callback was invoked due to programmatic edits.

- XtNmotionCallback

Class:	XtCCallback
Type:	XtCallbackList
Def:	NULL
Access:	SGIO
Action:	This callback is invoked when cursor position changes.

The calldata structure is:

```
OlTLMotionCallbackStruct {
  int    reason,
  XEvent  *event,
  Booleanvalid,
  int    current_cursor,
  int    new_cursor
}
```

Fields in the calldata structure are:

reason	OL_REASON_MOTION or OL_REASON_PROG_MOTION
event	corresponding XEvent structure
valid	If True, widget performs the Cursor motion, else doesn't.
current_cursor	Current position of cursor.
new_cursor	New cursor position.

Note – OL_REASON_MOTION indicates that the callback was invoked due to non-programmatic cursor-movement. OL_REASON_PROG_MOTION indicates that this callback was invoked due to programmatic cursor-movement.

- `XtNcommitCallback`

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGIO
Action:	This callback is invoked when a <TAB> or <Return> is inserted into the TextLine.

The calldata structure is:

```
OlTLCommitCallbackStruct {
 int   reason,
 XEvent  *event,
 Booleanvalid,
 OlStrbuffer
 int   length
}
```

Valid fields in the calldata structure are:

`reason`	`OL_REASON_COMMIT`
`event`	corresponding `XEvent` structure
`valid`	Explanation follows....
`buffer`	The text keyed in so far. Note this is widget-local storage.
`length`	Length of text

This callback would be invoked on <TAB> (if `XtNinsertTab` is `False`) or <Return>. If the callback sets "valid" to `True`, the widget transfers focus to the next traversable widget. If mouseless is ON, transfer focus and insert-point to next `TextLine`/Edit if mouseless is OFF. If the callback sets "valid" to `False`, the widget maintains focus and insert-point in the current `TextLine`. The application can provide additional feedback within this callback - like Popping up a Notice, clearing the field, resetting the cursor position, etc.

Additional Resources for Caption Support

The `TextLine` provides the following resources to support captions. However, note that these resources are subject to change in the next release. Also note that these resources are not settable.

- XtNcaptionPosition

Class:	XtCCaptionPosition
Type:	OlDefine
Def:	OL_LEFT
Access:	G
Action:	Specifies the position of the caption with regard to the text area.

- XtNcaptionWidth

Class:	XtCCaptionWidth
Type:	Dimension
Def:	0
Access:	G
Action:	Width of caption text in pixels.

- XtNcaptionAlignment

Class:	XtCCaptionAlignment
Type:	OlDefine
Def:	OL_CENTER
Access:	G
Action:	Specifies the alignment of the caption with regard to the text area.

- XtNcaptionSpace

Class:	XtCCaptionSpace
Type:	Dimension
Def:	4
Access:	G
Action:	Separation between the caption and the text area in pixels.

Convenience Functions

- Cursor Related

The following function can be used to get some key positions:

```
int OlTLGetPosition(Widget w, int pos)
```

Valid values for "pos" are OL_CURSORPOS, OL_BEGIN_CURRENT_WORD, OL_END_CURRENT_WORD, OL_END_LINE

- Text Setting and retrieval
  - Get a substring:
    `OlStr OlTLGetSubString (Widget w, int start, int length)` - returns "length" characters starting from "start". Caller is responsible for freeing the returned pointer. Returns `NULL` on failure. Failure includes invalid "start" and "length" values.
  - Insert a substring:

```
Boolean OlTLSetSubString(Widget w, int start, int length, OlStr buffer)
```

    Replaces "length" characters, beginning at "start" with "buffer". This function can be used to do Insertion, Replacement and Deletion; it returns `True` on success, `False` on failure. Failure includes invalid "start" and "length" values and memory allocation failures.

- Selection

```
Boolean OlTLSetSelection (Widget w, int start, int length)
```

  Selects "length" characters beginning at "start". Returns `True` on success, `False` on failure. Failure includes invalid start and length values.

```
OlStr OlTLGetSelection (Widget w, int *start, int *length )
```

  Returns the current selection as well as the selection start and length. Returns `NULL`, if no selection is active. The returned string should be freed by the application, when no longer required using `XtFree()`.

```
Boolean OlTLOperateOnSelection(Widget w, int mode)
```

Valid values for mode are:

`OL_CUT`	Copies the primary selection to the clipboard; then deletes the primary selection. Invokes the Pre and Post modifyVerification callbacks
`OL_COPY`	Copies the primary selection to the clipboard.
`OL_PASTE`	Inserts the clipboard selection at the destination cursor. If the destination cursor is inside the current selection, the clipboard selection replaces the selected text. Invokes the Pre and Post modifyVerification callbacks.
`OL_CANCEL`	Clears the primary selection.

Returns `False` if the primary selection is NULL, if the widget doesn't own the primary selection or if the function is unable to gain ownership of the clipboard selection, else returns `True`.

Activation Types

See the *OLIT Reference Manual* for documentation.

NumericField Widget

Synopsis

```
#include <Xol/OpenLook.h>
#include <Xol/NumericFie.h>
```

Class

Name: `NumericField`

Pointer: `numericFieldWidgetClass`

Ancestry

Core -> Primitive -> TextLine -> NumericField

Description

The `NumericField` widget is a one-line input field for alphanumeric text. The `NumericField` implements the OPEN LOOK Numeric TextField object. The widget supports the basic datatypes—integers and floats—and provides hooks for more complicated user-defined datatypes. Providing support for any datatype, basically involves:

- Implementing Converters between the datatype and String.
- Implementing per-key and per-field validation.
- Implementing increment and decrement operations.

Once the input-focus is moved into the widget, keyboard entry is allowed. If the input value exceeds the length of the input field, the ScrollButtons appear. Hidden text can then be scrolled into view, by pressing the ScrollButtons. Pressing the buttons continuously, scrolls the text repeatedly with a user-adjustable delay. The Increment or Decrement buttons can be used to increment or decrement the input value in an appropriate manner.

The `NumericField` contains the following graphical elements:

- Right justified bold label at the left of the input field
- Input field
- Input Caret (not present in read-only mode)
- 1 point (for Mono) or chiseled underline (not present in read-only mode)
- Optional ScrollButtons
- Increment and Decrement buttons referred to as DeltaButtons hereafter

Keyboard Traversal

The NumericField allows keyboard entry if it is sensitive and it has the keyboard focus. However in ReadOnly mode, the widget is not traversable and it does not receive keyboard focus. Selection is also turned off during ReadOnly mode.

The NumericField responds to the following keyboard navigation keys:

- NEXT_FIELD moves to the next traversable widget in the window
- PREV_FIELD moves to the previous traversable widget in the window
- CHARFWD moves the caret forward one character
- CHARBAK moves the caret backward one character
- WORDFWD moves the caret forward one word
- WORDBAK moves the caret back one word
- LINESTART moves the caret to the beginning of the display
- LINEEND moves the caret to the end of the display
- MENUKEY posts the menu associated with the NumericField

The NumericField responds to the following edit keys:

- DELCHARFWD deletes the character to the right of the caret
- DELCHARBAK deletes the character to the left of the caret
- DELWORDFWD deletes the word to the right of the caret
- DELWORDBAK deletes the word to the left of the caret
- DELLINEFWD deletes to the end of the line from the caret
- DELLINEBAK deletes from the beginning of the line to the caret
- DELLINE deletes the line containing the caret
- UNDO undoes the last edit

Display of the Keyboard Mnemonic

The NumericField does not display any mnemonic. If the NumericField is the child of a Caption widget, the caption can be used to display any mnemonic.

Display of the Keyboard Accelerator

The NumericField does not respond to any keyboard accelerators.

Display of Text

The NumericField displays its contents in the font specified by the XtNfont resource. If the length of the text exceeds the length of the input field, the widget sets up the Left or Right or both ScrollButtons to indicate this. The text is then visually truncated at the boundaries to show only as many characters as can fit in the input field. The truncation is always at a character boundary. A ScrollButton is present only if characters are hidden in the direction indicated by that buttons

The user can scroll to show the hidden parts of the text by clicking or pressing the ScrollButtons. Clicking SELECT on any ScrollButton will scroll the text one character in the direction indicated by that button. Pressing SELECT on any ScrollButton will repeat the scrolling with a user-adjustable delay between each scroll.

Caret position

As characters are entered from the keyboard, the caret moves to the right until it reaches the right end of the input field. As additional characters are typed, the text jump-scrolls to the left by a specific amount.

Note – The `NumericField` always keeps the cursor visible. Thus the presence or absence of either of the ScrollButtons is controlled by the current cursor position.

Selection Of Text

Text selection can be done by the user by using the mouse or keyboard. The widget also provides a set of convenience functions to manipulate the selection programmatically.

DeltaButtons

The DeltaButtons can be used to increment or decrement the value in the `NumericField` by a specified amount. The widget provides default logic for the supported datatypes (Ints and floats) as well as hooks for user-defined datatypes. The DeltaButtons become insensitive when the value equals any set value-limits.

Coloration

See SnapShot of this feature in OpenWindows chapter.

Widget Resources

Table 16-1 NumericField SuperClass Resources - Core

Name	Type	Default	Access
XtNx	Position	0	SGI
XtNy	Position	0	SGI
XtNwidth	Dimension	<computed>	SG
XtNheight	Dimension	<computed>	SG
XtNborderWidth	Dimension	0	SGI
XtNsensitive	Boolean	True	GIO
XtNancestorSensitive	Boolean	True	G
XtNscreen	Screen *	parent's	GI
XtNdepth	Cardinal	parent's	GI
XtNcolormap	Colormap	parent's	GI
XtNbackground	Pixel	parent's	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGI
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNmappedWhenMananaged	Boolean	True	SGI
XtNdestroyCallback	XtCallbackList	NULL	SGIO

Table 16-2 NumericField SuperClass Resources - Primitive

Name	Type	Default	Access
XtNaccelerator	XtRString	NULL	U
XtNacceleratorText	XtRString	NULL	U
XtNconsumeEvent	XtRCallback	NULL	SGIO
XtNfont	XtROlFont	XtDefaultFont	SGI
XtNfontColor	XtRPixel	XtDefaultForeground	SGID
XtNforeground	XtRPixel	XtDefaultForeground	SGID
XtNinputFocusColor	XtRPixel	"Red"	SGI
XtNmnemonic	OlRChar	NULL	SGI
XtNreferenceName	XtRString	NULL	SGI
XtNreferenceWidget	XtRWidget	NULL	SGI
XtNtextFormat	XtROlStrRep	?	?
XtNtraversalOn	Boolean	True	SGI
XtNuserData	XtPointer	NULL	SGI

Table 16-3 NumericField SuperClass Resources - TextLine

Name	Type	Default	Access
XtNblinkRate	Int	1000	SGI
XtNcaptionLabel	OlStr	NULL	SGI
XtNcaptionFont	OlFont	OlDefaultBoldFont	SI
XtNcharsVisible	Int	0	GI
XtNinitialDelay	Int	500	SGI
XtNrepeatRate	Int	100	SGI
XtNinsertTab	Bool	False	SGI
XtNmaximumChars	Int	0	GI
XtNstring	OlStr	NULL	SGI
XtNcursorPosition	Int	0	SGI
XtNeditType	OlDefine	OL_TEXT_EDIT	SGI
XtNupdateDisplay	Boolean	True	SGI
XtNunderline	Boolean	True	
XtNmenu	Widget	NULL	GI
XtNpreModifyCallback	XtCallbackList	<Special>	SGIO
XtNpostModifyCallback	XtCallbackList	NULL	SGIO
XtNmotionCallback	XtCallbackList	NULL	SGIO
XtNcommitCallback	XtCallbackList	<Special>	SGIO

Table 16-4 Additional Resources for Caption Support

XtNcaptionPosition	OlDefine	OL_LEFT	G
XtNcaptionWidth	Dimension	0	G
XtNcaptionAlignment	OlDefine	OL_CENTER	G
XtNcaptionSpace	Dimension	4	G

Table 16-5 TextLine Convenience Functions for NumericField Widget

Int	OlTLGetPosition(Widget w, int pos)
OlStr	OlTLGetSubString(Widget w, int start, int length)
Boolean	OlTLSetSubString(Widget w, int start, int length, OlStr buffer)
Boolean	OlTLSetSelection (Widget w, int start, int length)
OlStr	OlTLGetSelection (Widget w, int *start, int *length)
Boolean	OlTLOperateOnSelection(Widget w, int mode)

NumericField Resources

The `NumericField` widget supports the basic datatypes: `int` and `float`. The widget also provides a mechanism for you to implement your own datatypes. Since the widget can support multiple datatypes, all resources which refer to any data are uniformly represented as pointers to the actual data - which could be any of the above widget supported types or the programmer's specific type. Implementing additional datatypes will be discussed in a later section.

- `XtNvalue`

Class:	`XtCValue`
Type:	`XtPointer`
Def:	`NULL`
Access:	SGI
Action:	Specifies a pointer to an `int` or `float` or the user-defined datatype. The contents of the pointer would be the data to be displayed. The data is then converted into a string (depending on the datatype) and used as the `XtNstring` resource of the superclass - the `TextLine` widget. By default, the widget makes internal copies of the data. Setting the `XtNsizeOf` resource to `0` prevents the widget from doing so. Refer the section on "Implementing New DataTypes" for reasons and techniques of doing this.
Values:	A pointer to valid data in the appropriate format.

Example for `XtSetValues`:

```
{
 int i = 10;
 XtVaSetValues(w, XtNvalue, &i, NULL);
}
```

Example for `XtGetValues`:

```
{
 int i = *ip;
 XtVaGetValues(w, XtNvalue, &ip, NULL);
}
```

- XtNtype

Class:	`XtCType`
Type:	`String`
Def:	`XtRInt`

Access:	IG
Action:	Specifies the type of the data.
Values:	XtRInt: Integer XtRFloat: Float Programmers can also define their own datatype, which will be described later.

- XtNsizeOf

Class:	XtCSizeOf
Type:	Cardinal
Def:	<Computed internally by widget>
Access:	IG
Action:	Specifies the size of the datatype that is used by the program. This value needs to be set only when the programmer has defined his/her own datatype. The special value of 0 can be used to indicate that the widget need not make copies of any resources. Refer to the section on “Implementing New Datatypes" for reasons and techniques.
Values:	Size of the datatype or 0

- XtNdelta

Class:	XtCDelta
Type:	XtPointer
Def:	1 for Ints and 0.1 for Floats; else NULL
Access:	SGI
Action:	Specifies the increment or decrement delta to be applied to the data field, when either of the DeltaButtons is pressed. The widget applies the delta internally for ints and floats. By default, the widget makes internal copies of the data. Setting the XtNsizeOf resource to 0 prevents the widget from doing so. Refer to the section on "Implementing New Datatypes" for reasons and techniques of.
Values:	A pointer to a valid data in the appropriate format.

- XtNdeltaState

Class:	XtCDeltaState
Type:	OlDefine
Def:	OL_ACTIVE

Access:	SGI
Action:	Specifies the state of the DeltaButtons. The possible values are:
Values:	`OL_ACTIVE`/"active": Both buttons are active `OL_INCR_INACTIVE`/"incr_inactive": Increment button is inactive; Decrement button is active. `OL_DECR_INACTIVE`/"decr_inactive": Decrement button is inactive; Increment button is active. `OL_ABSENT`/"absent": DeltaButtons are not present.

- `XtNmaxValue`

Class:	`XtCMaxValue`
Type:	`XtPointer`
Def:	`NULL`
Access:	SGI
Action:	Specifies the maximum value that can be set for the data field. In addition, for datatypes: `XtRInt` and `XtRFloat`, the widget does the following if this resource is not NULL: (a) While incrementing the data-field, if the value exceeds/equals `XtNmaxValue`, the incr-button is desensitized. (b) When the value is committed by the user, the widget verifies that the value does not exceed `XtNmaxValue`. By default, the widget makes internal copies of the data. Setting the `XtNsizeOf` resource to `0` prevents the widget from doing so. Refer to the section on "Implementing New Datatypes" for reasons and techniques.
Values:	A pointer to a valid data in the appropriate format.

- `XtNminValue`

Class:	`XtCMinValue`
Type:	`XtPointer`
Def:	`NULL`
Access:	SGI
Action:	Specifies the minimum value that can be set for the data field. In addition, for datatypes: `XtRInt` and `XtRFloat`, the widget does the following if this resource is not NULL:

	(a) During decrementing the value-field, if the value equals or is less than `XtNminValue`, the decr-button is desensitized. (b) When the value is committed by the user, the widget verifies that the value is not less than `XtNminValue`. By default, the widget makes internal copies of the data. Setting the `XtNsizeOf` resource to `0` prevents the widget from doing so. Refer to the section on "Implementing New Datatypes" for reasons and techniques.
Values:	A pointer to a valid data in the appropriate format.

- `XtNdeltaCallback`

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGIO

Callback structure:

```
OlNFDeltaCallbackStruct {
int              reason;/* READ_ONLY */
XEvent *event; /* READ_ONLY */
XtPointerdelta;/* READ_ONLY */
XtPointercurrent_value;/* READ_ONLY */
OlDefinecurrent_delta_state; /* READ_ONLY */
XtPointernew_value;
OlDefinenew_delta_state;
Booleanupdate;
}
```

Note – This callback is invoked when either of DeltaButtons are pressed. If the buttons are pressed continously, the callback will be invoked multiple times. Also note that the callback would obviously not be invoked if the DeltaButtons are not present.

The fields signify:

`reason`	`OL_REASON_INCREMENT` / `OL_REASON_DECREMENT`
`event`	Corresponding `XEvent` structure
`delta`	The value of the `XtNdelta` resource.
`current_value`	The current value of the `XtNvalue` resource.
`current_delta_state`	The current value of `XtNdeltaState`.

new_value	The new value of the XtNvalue resource if the widget's default delta logic (if any) is carried out. The programmer can override the default behavior by changing the contents of this field to any new value.
new_delta_state	The new value of the XtNdeltaState if the widget's default delta logic (if any) is carried out. The programmer can override the default behavior by changing this field to any new value.
update	This resource should be set to True to indicate that the "new_value" has changed.

a. If XtNtype is user defined, then modifying "new_value" to the new value AND setting "update" to True, would cause the widget to update XtNvalue with the new value AND erase the old_value and display the new_value.

If the value of the "new_delta_state" field changes, the widget would update the XtNdeltaState resource and redisplay the DeltaButtons appropriately.

b. If XtNtype is XtRFloat/XtRInt, the "new_value field" is computed by the widget by incrementing/decrementing XtNvalue by XtNdelta and validating it against XtNmaxValue/XtNminValue. XtNdeltaState is also set up appropriately by the widget.

The programmer can override this default logic by modifying "new_value" AND setting "update" to True. This would cause the widget to update XtNvalue with the new value, erase the old_value and display the new_value. Note that no increment/decrement OR validation will be done by the widget.

If the value of the "new_delta_state" field is changed, the widget would update the XtNdeltaState resource and redisplay the DeltaButtons appropriately.

Note – This is also the default behavior, if NO callbacks are registered.

- XtNvalidateCallback

Class:	XtCCallback
Type:	XtCallbackList
Def:	NULL
Access:	SGIO

Calldata structure:

```
OlNFValidateCallbackStruct {
  int reason;
  XEvent *event;
  XtPointervalue;
```

```
OlDefinedelta_state;
Boolean  update;
Boolean valid;
}
```

The callback is invoked on <TAB>, <CR> or <FocusOut>. The fields signify:

reason	OL_REASON_VALIDATE
event	Corresponding XEvent structure
value	The current value of the XtNvalue resource. The user can make appropriate changes to this field and set the "update" flag to update XtNvalue.
delta_state	The current value of XtNdeltaState. This can be set to a new value to update this resource.
update	This resource should be set to True to indicate that the "value" has changed.
valid	See below.

If XtNtype is XtRInt or XtRFloat, then the widget would internally validate value against XtNmaxValue and XtNminValue. If the value is invalid, it sets "valid" to False. The callback can take the appropriate action , such as beeping, popping up a notice, etc. It can also reset the "value" and "delta_state" resources to suitable default values by modifying those fields and setting "update" to True.

On return from the callback, if "valid" is False, the widget maintains focus within the widget. If "valid" is True, the widget relinquishes focus.

If XtNtype is user defined, the widget invokes this callback right away. Depending on the "valid" flag, the widget would maintain focus or relinquish focus.

Per-Key and Per-Field Validation

The NumericField widget uses the callbacks of its superclass to implement per-key and per-field validation as described below.

The NumericField widget itself does per-key validation for datatypes XtRInt and XtRFloat. Basically, the widget registers appropriate XtNpreModifyCallbacks on its superclass (TextLine) for ints and floats. The callback for int verifies that each keystroke is a valid digit; the callback for float ensures that each keystroke is either a digit or a decimal-separator and that only one decimal-separator occurs in the field. The programmer can override this behavior by removing the widget-registered callback and installing his/her own callback.

For user-defined datatypes, the programmer can register his/her own per-key validation callback.

The NumericField widget also does per-field validation for datatypes XtRInt and XtRFloat which verifies that the field-value is within the min and max ranges. Basically, the widget registers a callback on the XtNcommitCallback of the TextLine superclass. This callback verifies the field-value against the min and max ranges, if they exist.

On success, the widget sets the valid field of the OlNFValidateCallbackStruct to True and invokes any XtNvalidateCallback callbacks registered. On failure, the valid flag is set to False and any XtNvalidateCallback callbacks present are invoked.

Implementing New Datatypes

The NumericField widget currently supports Integer and Float as the base datatypes. i.e., the widget internally provides all logic and code necessary to handle these datatypes. The widget also allows programmers to define and use their own datatypes. Examples of such user-defined datatypes would include Date, Time, Currency, etc.

Implementing a new datatype involves the following steps:

1. Register the new datatype with the widget.
2. Implement Converters between the datatype and String.
3. Optionally implement PerKey and PerField Validation routines.
4. Optionally implement Increment and Decrement operations.

Each step is elaborated below:

1. The programmer should specify a name for the new datatype. The name should be a printable ASCII string. This name is registered with the widget by setting it up as the XtNtype resource.

   The programmer should also set the XtNsizeOf resource to the size in bytes of the new datatype. The widget internally uses this value, while making copies of the contents of XtNvalue, XtNdelta, XtNminValue and XtNmaxValue. XtNsizeOf can be set to the special value of '0' to indicate that the widget need not make copies of the above resources. This will result in the widget accessing these values directly through pointers to the application's dataspace. Typically, a programmer would want to do this if the datatype is large and the application stores all the above resource values in malloced or global space, so that copying between the application's dataspace and the widget's dataspace is redundant or inefficient. Caution should be used, however, since modifying any of these values would directly affect the widget's state too.

2. The `NumericField` displays the contents of `XtNvalue` by converting the contents to a string and setting it up as the `XtNstring` resource of its superclass, the `TextLine`. Similarly, the widget updates `XtNvalue` by retrieving the `XtNstring` resource value of the `TextLine` and converting it back to the specified datatype.

   The widget does these conversions internally for the base datatypes. However, for user-defined datatypes, the programmer needs to provide this conversion logic. This is done by setting up the `XtNconvertProc` function. The prototype of this function is as follows:

```
Boolean
Converter(Widget w, OlStrRep format, XrmQuark from_type, XrmQuark to_type,
XtPointer *value, XtPointer *string, Cardinal *string_length)
```

`w`	Specifies the widget.
`format`	Specifies the text_format of the widget.
`from_type`	Specifies the source datatype, i.e., the datatype to be converted from. Note that this is a quarkified representation of the datatype name.
`to_type`	Specifies the destination datatype, i.e., the datatype to be converted to. Note that this is a quarkified representation of the datatype name.
`value`	Specifies a pointer to the data stored.
`string`	Specifies a pointer to the data stored as a string.
`string_length`	Specifies the size of the memory where the string is stored.

The Converter should do the following:

- Attempt the type conversion from the source datatype to the destination datatype.
- Copy the converted data to the appropriate location, i.e., in "string" if the destination datatype is string; in "value" if the destination datatype is the user's datatype. Before copying, ensure that the location is a valid memory pointer and that the storage-size is sufficient. If not, allocate sufficient memory and then copy the data. Update `*string_length` to the length of the string if the destination datatype is string.
- If the conversion failed, issue a warning and return `False`, else return `True`.

Note – The string datatype is identified as `XtRString`.

---

3. Per-Key validation ensures that each key is valid in the current context. The programmer can verify this by registering an appropriate `XtNpreModifyCallback` on the widget.

Per-Field validation ensures that the contents of the widget are valid, when the end-user commits a value (i.e., hits <return> or <Tab>). The programmer can achieve this by registering an appropriate `XtNvalidateCallback`. The callback should set the "valid" field of the calldata to indicate the validity of the data. The callback can also reset the widget contents by updating "value" and setting "update" to `True`.

4. The widget should update its contents and the state of its DeltaButtons whenever the buttons are pressed. The widget does this internally for the base datatypes. For user-defined datatypes, the programmer can implement the necessary logic by registering an appropriate `XtNdeltaCallback`. The callback should update the "`new_value`" and "`new_delta_state`" fields in the calldata and set "`update`" to `True`, if the contents of the widget need to be changed.

Activation Types

See the *OLIT Reference Manual* for documentation.

FileChooser Widget

FileChooser Widget Usage

The `FileChooser` widget allows the end-user to traverse directories (folders), view the files and subdirectories (folders) in them, and to select files.

Typically, an application uses a popup window file chooser, though it may use a file chooser panel embedded in a multi-category property sheet, hence the two widget classes for the OLIT file chooser implementation (see Synopsis).

The OLIT `FileChooserShell` widget class implements the OPEN LOOK file chooser user interface Object as specified in the functional specification currently being developed by the Human Interface Engineering (HIE) group.

The look and feel of this widget has three variants, depending on the operation it is intended to perform:

- Open
- Save
- Save As

Include a generic command, the Do Command.

These variants are implemented as a single modal widget class (see `XtNoperation`).

Like the spot help widget, only a single instance of a particular file chooser variant is allowed per application. However, the API does not attempt to enforce this user interface policy. The API specification which follows provides for:

- features beyond those in the Human Interface Engineering (HIE) specification
- a default behavior which complies with the HIE specification
- the flexibility to depart from the prescriptions of the HIE specification

Widget Architecture and the API

The API of the `FileChooser` widget classes is a pure resource-based Intrinsics API. There are no “convenience functions" in this API.

The functionality of the file chooser is available in two variants: a panel and a popup containing that panel. Hence the two widget classes, `FileChooser` and `FileChooserShell` (analogous to `XmFileSelectionBox` and `XmFileSelectionDialog`).

When a `FileChooser` is instantiated using the default values of its resources, the resulting UI object will be OPEN LOOK conformant.

The functional specification calls for no more then a single set of file chooser instances (`Open`, `Save`, `Save As`) to be created and made available to the end user. It is the responsibility of the application to enforce this policy.

FileChooser Widget Synopsis

```
#include <Xol/FileCh.h>
```

Class names: `FileChooser`

Class pointers: `fileChooserWidgetClass`

FileChooser Widget Resources

The widget resources provided by `FileChooser` fall into three categories:

- Base Resources
  - I18N
  - State parameters
  - Standard callbacks (and associated callback structures)
  - File filtering
  - Goto control

- Customization Resources
  - Sorting
  - Pathname processing
  - End-user feedback
  - End-user customization: accelerators and mnemonics
- Extensibility Resources
  These resources are provided for application programmers wishing to extend the `FileChooser`. They fall in these groups:
  - Extension container
  - Component access
  - Extensibility callbacks
  - Control of labels

Base Resources

STATE PARAMETERS

Name	Class	Type	Default	Access
XtNtextFormat	XtCTextFormat	OlStrRep	_OlGetDefaultTextFormat	IG

Determine the widget-wide international text encoding.

The toolkit computes the default setting. This should be removed if this resource becomes available in Manager.

Note – The values of the all messages are stored in string catalogs for localization purposes, including all status, error, and state messages.

Name	Class	Type	Default	Access
XtNfont	XtCFont	OlFont	XtDefaultFont	ISG
XtNfontColor	XtCFontColor	Pixel	XtDefaultForeground	ISG
XtNforeground	XtCForeground	Pixel	XtDefaultForeground	ISG
XtNscale	XtCScale	int	(shell's scale) or OL_DEFAULT_POINT_SIZE	ISG
XtNoperation	XtCOperation	OlDefine	OL_OPEN	IG

The operation to be supported by this `FileChooser` instance, one of:

- `OL_OPEN`/"open"
- `OL_SAVE`/"save"
- `OL_SAVE_AS`/"save_as"
- `OL_INCLUDE`/"include"
- `OL_DO_COMMAND`/"do_command"

The first four are the standard operations provided by the OPEN LOOK specification. The fourth, `OL_DO_COMMAND`/”do_command", which stands for generic command, is a general purpose mode to accommodate operations which cannot be implemented using the behavior of the behavior enforced by the first four operations.

Unlike the other operations, `do_command` mode allows both:

- selection of either folders or documents from the scrolling list, *and*
- selection of a document using either the scrolling list *or* the document type in field.

Name	Class	Type	Default	Access
`XtNcurrentFolder`	`XtCFolderName`	`String`	. (current working directory)	ISG

The argument to `XtNcurrentFolder` is the directory name to display, or the one being displayed as a result of the end-user's navigation with the file chooser.

- Relative and absolute fully-specified pathname (i.e., no wild cards, no filename or tilde expansion, no shell variable substitution); ‘.' and ‘..' are accepted.
- The default is to display the current working directory, which for many applications is the directory from which the application was invoked.
- Every time this resource is set, the file list view is updated.
- Setting this resource also causes `XtNopenFolderCallback` to be called.

While the OPEN LOOK functional specification calls for the user's home directory to be the file chooser's default initial folder, most applications will choose to set this resource to the user's current working directory, that is ".".

Name	Class	Type	Default	Access
`XtNlastDocumentName`	`XtCDocumentName`	`String`	`NULL`	ISG

This resource enables the application to provide the default root of the base name to be used in the `Save As` type-in field.

Name	Class	Type	Default	Access
XtNlistVisibleItemMinCount	XtCVisibleItemMinCou nt	Cardinal	3	IG
XtNlistVisibleItemCount	XtCVisibleItemCount	Cardinal	10	IG

This resource specifies the preferred number of files and folders to show in the list. This value must be positive. The list will always show at least a total of `XtNlistVisibleItemCountMin` entries.

Name	Class	Type	Default	Access
XtNshowGlyphs	XtCBooleanDefault	Boolean	True	ISG
XtNfollowSymlinks		Boolean	True	ISG

If a folder is visited via a symbolic link, then this resource would enable traversing the parent of the symbolic link rather that of the actual folder when a "Go up one folder" command is executed.

Name	Class	Type	Default	Access
XtNnoTypeInAcceleration	XtCBooleanDefault	Boolean	False	ISG
XtNallowIncrementalSearch	XtCBooleanDefault	Boolean	True	ISG

Standard Callbacks for the FileChooser Widget

The following are examples of standard callbacks used with the `FileChooser` widget.

```
typedef struct _OlFileChGenericCallbackStruct {
  /* OLIT standard fields */
  int      reason;/* OL_REASON_FOLDER_OPENED, ... */

  /* FileChooser standard fields */
  XtVersionTypeversion;
  XtPointerextension;/* reserved */
  OlDefineoperation;/* OL_OPEN, ... */
  String   current_folder;/* absolute pathname */

} OlFileChGenericCallbackStruct;
```

Name	Class	Type	Default	Access
XtNopenFolderCallback	XtCCallback	XtCallbackList	NULL	OSG

```
typedef struct _OlFileChFolderCallbackStruct {

  /* OLIT standard fields */
  int      reason; /* OL_REASON_OPEN_FOLDER */

  /* FileChooser standard fields */
  XtVersionTypeversion;
  XtPointerextension;/* reserved */
  OlDefineoperation;/* OL_OPEN, ... */
  String   current_folder;/* absolute pathname */

  /* Callback-dependent fields */
  String   request_folder;/* may be set to NULL */
  OlFNavNoderequest_folder_node;

} OlFileChFold
erCallbackStruct;
```

- `OL_REASON_OPEN_FOLDER`

  For all operations, this callback is invoked when:
  - a folder item in the list has been selected and the Open or Open Folder button is depressed, *or*
  - a folder item in the list is double-clicked, *or*
  - `XtNnoTypeInAcceleration` is set to `False` and a pathname expression which resolves to a unique folder pathname is entered into the Goto type-in (either by following it with an `OL_ENTER` action or by depressing the Goto button), *or*
  - an item is selected in the Goto Menu, *or*
  - a setting of the `XtNcurrentFolder` resource

  In a Save, Save As, and in a generic operation, this callback is invoked when:
  - `XtNnoTypeInAcceleration` is set to `False` and a pathname expression which resolves to a unique folder pathname is entered into the `Save` or `Save As` type-in (by following it with an `OL_ENTER` action).

In all cases, the client may override the requested directory change by manipulating the `request_folder` field of the callback structure. If the latter is set to NULL, the current directory remains unchanged. A successful change causes the value of `XtNcurrentFolder` to be updated.

Name	Class	Type	Default	Access
XtNinputDocumentCallback	XtCCallback	XtCallbackList	NULL	OSG

```
typedef struct _OlFileChDocumentCallbackStruct {
```

```
  /* OLIT standard fields */
  int      reason;  /* OL_REASON_INPUT_DOCUMENT */
                    /* OL_REASON_OUTPUT_DOCUMENT */

  /* FileChooser standard fields */
  XtVersionTypeversion;
  XtPointerextension;/* reserved */
  OlDefineoperation;/* OL_OPEN, ... */
  String   current_folder;/* absolute pathname */

  /* Callback-dependent fields */
  String   request_document_folder;
                    /* absolute pathname */
  String   request_document; /* base name */
  OlFNavNoderequest_document_node;

} OlFileChDocumentCallbackStruct;
```

- `OL_REASON_OPEN_DOCUMENT`

  In an Open or Include operation, this callback list is invoked when:
  - a document item in the list has been selected and the Open or Include button is depressed, *or*
  - a document item in the list is double-clicked, *or*
  - `XtNnoTypeInAcceleration` is set to `False` and a pathname expression which resolves to a unique document pathname is entered into the Goto type-in (by following it with an `OL_ENTER` action).

Name	Class	Type	Default	Access
XtNoutputDocumentCallback	XtCCallback	XtCallbackList	NULL	OSG

The list of callback functions to be called in when the Save or Save As command buttons are depressed.

OlFileChDocumentCallbackStruct

- `OL_REASON_OUTPUT_DOCUMENT`
  In a Save or Save As operation, this callback list is invoked when a name expression is entered into the Save or Save As type-in (by following it with an `OL_ENTER` action).

- File Filtering

Name	Class	Type	Default	Access
XtNfilterString	XtCFilterString	String	NULL	ISG

An ed(1)-like regular expression string used to filter document names from the list (except if they are directory names). Those files which are filter will be displayed as inactive (‘grayed-out') if the XtNshowInactive resource is set to True, and will not be displayed at all if that resource is set to False.

The expression “ .*” causes the XtNfilterProc to be called for all files in the list. No callbacks are performed if the expression is empty ("") or NULL.

Name	Class	Type	Default	Access
XtNfilterProc	XtCFilterProc	XtCallbackProc	NULL	O

```
typedef struct _OlFiChFilterCallbackStruct {

  /* OLIT standard fields */
  int      reason; /* OL_REASON_FILTER */
  XEvent*  event;  /* NULL */

  /* FileChooser standard fields */
  XtVersionTypeversion;
  XtPointerextension;/* reserved */
  XtPointeruser_data;/* application-specified */
  OlDefineoperation;/* OL_OPEN, ... */
  String   current_folder;/* absolute pathname */

  /* Callback-dependent fields */
  String   filtered_document;/* base name */
  struct stat*filtered_document_stat_buffer;

  Boolean  is_active;
               /* TRUE unless overriden by client */

  XImage*  filtered_document_glyph;

} OlFiChFilterCallbackStruct;
```

- OL_REASON_FILTER

The callback called for each filtered (i.e., matching XtNfilterString) file in the list. The callback allows the application to specify custom glyphs for filtered files.

Name	Class	Type	Default	Access
XtNshowInactive	XtCBooleanDefault	Boolean	True	ISG

Specifies the display of filtered out entries in the list. If enabled, they are shown as inactive (grayed-out).

Name	Class	Type	Default	Access
`XtNhideDotFiles`	`XtCBooleanDefault`	`Boolean`	`True`	ISG

Specifies if files with names beginning with a dot should be displayed in the list.

- `GoTo`

Name	Class	Type	Default	Access
`XtNhomeFolder`	`XtCHomeFolder`	`String`	(home of uid)	ISG
`XtNapplicationFoldersMaxCount`	`XtCFolderMaxCount`	`Cardinal`	`5`	IG

The maximum allowed number of application folders. The actual count of these folders may not exceed this value and is determined by the value of the `XtNapplicationFolders` list.

Name	Class	Type	Default	Access
`XtNapplicationFolders`	`XtCFolders`	`OlFolderList`	`NULL`	ISG

```
typedef String*OlFolderList;
```

Items in the list exceeding the value of `XtNapplicationFoldersMaxCount` are ignored. The syntax of the Intrinsics Translation Manager is used. For example,

```
"~/folder1\n
 ~/folder2\n
 /folder/subfolder"
```

Name	Class	Type	Default	Access
`XtNuserFoldersMaxCount`	`XtCFolderMaxCount`	`Cardinal`	`5`	IG

The maximum allowed number of user folders. The actual count of these folders may not exceed this value and is determined by the value of the `XtNuserFolders` list.

Name	Class	Type	Default	Access
`XtNuserFolders`	`XtCFolders`	`OlFolderList`	`NULL`	ISG

Items in the list exceeding the value of `XtNuserFoldersMaxCount` are ignored. The syntax of the Intrinsics Translation Manager is used. For example,

```
"~/folder1\n
 ~/folder2\n
```

```
  /folder/subfolder"
```

These resources are to be in synchronization with their equivalents in the workspace resources, when available.

Name	Class	Type	Default	Access
XtNhistoryFoldersMinCount	XtCFolderMinCount	Cardinal	3	IG
XtNhistoryFoldersMaxCount	XtCFolderMaxCount	Cardinal	15	IG

The maximum allowed number of user folders. The actual count of these folders may not exceed this value and is determined by the value of the `XtNhistoryFolders` list.

If the vale of `XtNhistoryFodlersMaxCount` is less than that of `XtNhistoryFoldersMinCount`, the former is reset to the latter.

Setting both of these resource to `0` (zero) disables the folder history mechanism.

Name	Class	Type	Default	Access
XtNhistoryFolders	XtCFolders	OlFolderList	NULL	G

Items in the list exceeding the value of `XtNhistoryFoldersMaxCount` are ignored. The syntax of the Intrinsics Translation Manager is used. For example,

```
 "~/folder1\n
  ~/folder2\n
  /folder/subfolder"
```

Customization Resources

Porting

Name	Type	Default	Access
XtNcomparisonFunc	XtCComparisonFunc	`OlComparisonFunc` (string case-insensitive ascending)	ISG

```
typedefint (*OlStrComparisonFunc)(
  const OlStr left_string, const OlStr right_string);

typedefint (*OlComparisonFunc)(
  const XtPointer left_key, const XtPointer right_key);
```

The string comparison function used to sort the directory entries in the list display.

OlStrComparisonFunc compares its arguments and returns an integer greater than, equal to, or less than 0, if left_string is lexicographically greater than, equal to, or less than right_string.

OlComparisonFunc compares its arguments and returns an integer greater than, equal to, or less than 0, if left_key is greater than, equal to, or less than right_key, where the meaning or order is application specified, as is the choice of key field within the unspecified structures of the arguments.

The following sort styles are provided by the toolkit and may be specified with these symbolic constants:

Sort Style	OlComparisonFunc
case-sensitive ascending	OL_SORT_STR_CASE_ASCENDING
case-sensitive descending	OL_SORT_STR_CASE_DESCENDING
case-insensitive ascending	OL_SORT_STR_NO_CASE_ASCENDING
case-insensitive descending	OL_SORT_STR_NO_CASE_DESCENDING

Note – By default, sorting of the file list is performed according to the collation sequence specified in the locale's LC_COLLATE.

Pathname Processing

Name	Type	Default	Access	
XtNexpandTilde	XtCBooleanDefault	Boolean	True	ISG
XtNsubstituteShellVariables	XtCBooleanDefault	Boolean	True	ISG

End User Customization

Table 16-6 Accelerators

Name	Class	Type	Default	Access
XtNgotoHomeAccelerator	XtCAccelerator	String	NULL	ISG
XtNgoUpOneFolderAccelerator	XtCAccelerator	String	NULL	ISG
XtNopenFolderAccelerator	XtCAccelerator	String	NULL	ISG
XtNcancelAccelerator	XtCAccelerator	String	NULL	ISG
XtNopenAccelerator	XtCAccelerator	String	NULL	ISG
XtNsaveAccelerator	XtCAccelerator	String	NULL	ISG
XtNsaveAsAccelerator	XtCAccelerator	String	NULL	ISG
XtNincludeAccelerator	XtCAccelerator	String	NULL	ISG
XtNcommandAccelerator	XtCAccelerator	String	NULL	ISG

The accelerator used for the Home entry in the Goto menu.

In OLIT, accelerators are restricted to 7-bit single byte.

Table 16-7 Mnemonics

Name	Class	Type	Default	Access
XtNgotoHomeMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNopenFolderMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNcancelMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNopenMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNsaveMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNsaveAsMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNincludeMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG
XtNcommandMnemonic	XtCMnemonic	OlMnemonic	'\0'	ISG

The mnemonic character used for the Home, Open, Open Folder, Save, Save As, and Cancel command buttons at the bottom of a file chooser, respectively. In OLIT, mnemonics are restricted to 7-bit single byte.

Extensibility Resources

- Extension Container

Name	Class	Type	Default	Access
XtNextensionName	XtCExtensionName	String	NULL	IG

The application-specified instance name of the extension container widget instance used to parent additional controls to be provided by the application.

In an Open operation, this container widget extends between the file scrolling list and the command buttons at the bottom of the file chooser panel. In a Save or Save As operation, this container widget extends between the Save type-in field and the command buttons at the bottom of the file chooser panel.

The container is created by the `FileChooser` widget on behalf of the application, if, and only if, this instance name is not NULL. This extension container is not intended to resize.

This instance name may be used for resource settings in a defaults file.

Name	Class	Type	Default	Access
`XtNextensionClass`	`XtCExtensionClass`	`WidgetClass`	(formWidgetClass)	IG

The class name requested by the application for the extension container widget.

Name	Class	Type	Default	Access
`XtNextensionWidget`	`XtCExtensionWidget`	`Widget`	`NULL`	G

The toolkit-provided widget ID of the extension container instance, if present.

Table 16-8 Component Access

Name	Class	Type	Default	Access
`XtNgotoPromptWidget`	`XtCComponentWidget`	`Widget`	(toolkit)	G
`XtNgotoButtonWidget`	`XtCComponentWidget`	`Widget`	(toolkit)	G
`XtNgotoMenuWidget`	`XtCComponentWidget`	`Widget`	(toolkit)	G
`XtNgotoHomeButtonWidget`	`XtCComponentWidget`	`Widget`	(toolkit)	G
`XtNgotoTypeInWidget`	`XtCComponentWidget`	`Widget`	(toolkit)	G

In an Open operation, this widget has the initial input focus.

Hence, an OL_DEFAULT action for the file chooser resolves to the verification callback of this field.

Name	Class	Type	Default	Access
XtNcurrentFolderLabelWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNcurrentFolderWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNlistPromptWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNdocumentListWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNdocumentNameLabelWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNdocumentNameTypeInWidget	XtCComponentWidget	Widget	(toolkit)	G

In a Save or Save As operation, this widget has the initial input focus. Hence, an OL_DEFAULT action for the FileChooser resolves to the verification callback of this field.

Name	Class	Type	Default	Access
XtNopenButtonWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNcancelButtonWidget	XtCComponentWidget	Widget	(toolkit)	G
XtNcommandButtonWidget	XtCComponentWidget	Widget	(toolkit)	G

These resources are used to access a component within a file chooser. They may be used by the client to unmanage unwanted components.

Note – An application should not assume that the returned widget ID will be of any particular class.

See also XtNextensionWidget above.

Extensibility Callbacks

These resources are provided to augment the standard internal call backs for the benefit of applications that need to extend the standard file chooser behavior.

Name	Class	Type	Default	Access
XtNlistChoiceCallback	XtCCallback	XtCallbackList	NULL	OSG

```
typedef struct _OlFileChListChoiceCallbackStruct {

  /* OLIT standard fields */
  int      reason; /* OL_REASON_LIST_CHOICE */

  /* FileChooser standard fields */
  XtVersionTypeversion;
  XtPointerextension;/* reserved */
  OlDefineoperation;/* OL_OPEN, ... */
  String   current_folder;/* absolute pathname */

  /* Callback-dependent fields */
  String   chosen_item;
  OlFNavNodechosen_item_node;

} OlFileChListChoiceCallbackStruct;
```

- `OL_REASON_LIST_CHOICE`

  The list of callback functions to be called in addition to the standard callback when an entry is selected in the list.

Name	Class	Type	Default	Access
XtNfolderOpenedCallback	XtCCallback	XtCallbackList	NULL	OSG

```
OlFileChGenericCallbackStruct
```

- OL_REASON_FOLDER_OPENED

  Determines the callback list after a directory change has been performed.

Name	Class	Type	Default	Access
XtNcancelCallback	XtCCallback	XtCallbackList	NULL	OSG

OlFileChGenericCallbackStruct

- OL_REASON_CANCEL

  In all operations, this callback list is invoked when the Cancel button is depressed.

Labels

The default values for the following resources relating to labels are stored in string catalogs for localization purposes.

These resources allow applications to override the default labels prescribed in the functional specification for the four standard operations (Open, Save, Save As, and Include). These resources will be modified to accomplish other operations not directly supported, for example: Print, Browse, or Compile.

Name	Class	Type	Default	Access
XtNgotoPromptString	XtCPromptString	OlStr	"Type in the path to the folder and press Return."	ISG
XtNgotoLabel	XtCLabel	OlStr	"Go To"	ISG
XtNgotoHomeLabel	XtCLabel	OlStr	"Home"	ISG
XtNcurrentFolderLabelString	XtCLabel	OlStr	"Current Folder:"	ISG

In all of the following four resources which specify message strings to be used in the file list prompt, the '%' (percent sign) character is used to signal the insertion position of the string for the name of the current operation.

The name of the operation is taken from the string used to label the button used to perform that operation in the file chooser panel. For example, in an OL_OPEN operation the _default_ label of the Save button is “Open”, and is the string used in deriving the list prompt. But, if the label had been customized to "Browse", the latter string would be used.

Hence, in the examples above the list prompt would be:

```
"Select a document or folder and click Open."
"Select a document or folder and click Browse."
```

See the specific resource for the specific button label mapping.

Name	Class	Type	Default	Access
XtNopenPromptString	XtCPromptString	OlStr	"Select a document or folder and click %."	ISG

This prompt string is used in the OL_OPEN operation only. '%' is replaced with label of Open button.

Name	Class	Type	Default	Access
XtNfolderPromptString	XtCPromptString	OlStr	"Select a folder and click %."	ISG

This prompt string is used in the OL_SAVE and OL_SAVE_AS operations. '%' is replaced with label of Open Folder button.

Name	Class	Type	Default	Access
XtNdocumentPromptString	XtCPromptString	OlStr	"Select a document and click %."	ISG

This prompt string is used in the OL_INCLUDE operation only. '%' is replaced with label of Include button.

Name	Class	Type	Default	Access
XtNcommandPromptString	XtCPromptString	OlStr	"Select a document and click % or a folder and click %."	ISG

This prompt string is used with the OL_DO_COMMAND (i.e., generic) operation only.

The first '%' is replaced with the label of the Do Command button, the second '%' is replaced with the label of the Open Folder button.

Name	Class	Type	Default	Access
XtNgoUpOneFolderLabel	XtCLabel	OlStr	"...Go up one folder..."	ISG
XtNopenFolderLabel	XtCLabel	OlStr	"...Go up one folder..."	ISG
XtNcancelLabel	XtCLabel	OlStr	"Cancel"	ISG
XtNopenLabel	XtCLabel	OlStr	"Open"	ISG
XtNsaveLabel	XtCLabel	OlStr	"Save"	ISG
XtNsaveAsLabel	XtCLabel	OlStr	"Save As"	ISG
XtNincludeLabel	XtCLabel	OlStr	"Include"	ISG
XtNcommandLabel	XtCLabel	OlStr	"Command"	ISG
XtNdefaultDocumentName	XtCDefaultDocumentName	OlStr	"Untitled1"	ISG
XtNdefaultDocumentSuffix	XtCDefaultDocumentSuffix	OlStr	".1"	ISG

FileChooserShell Widget

Synopsis

```
#include <Xol/FileChSh.h>
```

Class name: `FileChooserShell`

Class pointer: `fileChooserShellWidgetClass`

Notes

As called for by the GUI, the `FileChooserShell` derives its header from the title of the application, and the name of the operation that the file chooser operation is performing.

The title of the application is obtained from the main top level window of the application. This string may either have been provided by the application (e.g., "Foo Text Editor"), or, it may be the name of the executable used to invoke the application (e.g., "footextedit").

The name of the operation is taken from the string used to label the button used to perform that operation in the file chooser panel. For example, in a `OL_SAVE` operation the `_default_` label of the Save button is "`Save`", and is the string used in deriving the `FileChooserShell` header. But, if the label had been customized to "`Write`", the latter string would be used.

Hence in the examples above the header displayed would be one of:

- "Foo Text Editor: Save"
- "footextedit: Save"
- "Foo Text Editor: Write"
- "footextedit: Write"

Resources

- I18N

Name	Class	Type	Default	Access
`XtNtextFormat`	`XtCTextFormat`	`OlStrRep`	`_OlGetDefaultTextFormat`	IG

Determine the widget-wide international text encoding. The toolkit computes the default setting.

Note – The values of the all messages are stored in string catalogs for localization purposes, including: (1) all status, including error, messages, and (2) all state messages.

- State

Name	Class	Type	Default	Access
XtNoperation	XtCOperation	OlDefine	OL_OPEN	IG

The operation to be supported by this file chooser popup instance is one of:
- OL_OPEN/"open"
- OL_SAVE/"save"
- OL_SAVE_AS/"save_as"
- OL_INCLUDE/"include"
- OL_DO_COMMAND/"do_command"

Name	Class	Type	Default	Access
XtNpointerWarping	XtCPointerWarping	Boolean	OL_POINTER_WARPING	ISG

- Component Access

Name	Class	Type	Default	Access
XtNfileChooserWidget	XtCComponentWidget	Widget	(toolkit)	G

Name	Class	Type	Default	Access
XtNverifyCallback	XtCCallback	XtRCallback	NULL	OSG

```
typedef struct _OlFileChShVerifyCallbackStruct {

  /* OLIT standard fields */
  int      reason; /* OL_REASON_VERIFY */

  /* FileChooser standard fields */
  XtVersionTypeversion;
  XtPointerextension;/* reserved */
  OlDefineoperation;/* OL_OPEN, ... */
  String   current_folder;/* absolute pathname */

  /* Callback-dependent fields */
  Boolean  accept_verify;/* set to FALSE to reject */

} OlFileChShVerifyCallbackStruct;
```

- OL_REASON_VERIFY

Specifies the list of call back functions to be called when popping down a FileChooserShell widget. It allows interposing regardless of the cause of the pop down action (Open, Save, Save As, Cancel, Include, generic command, pin-out, dismiss).

Setting the accept_verify field to FALSE will defeat the pop down.

FontChooser Widget

Synopsis

```
#include <Xol/OpenLook.h>
#include <Xol/FontCh.h>
```

Class

Name: FontChooser

Pointer: fontChooserWidgetClass

Ancestry

Core -> Composite -> Constraint -> Manager -> RubberTile -> FontChooser

Description:

The FontChooser widget provides a user interface to conveniently choose fonts in an OLIT application. The widget does this by presenting lists of meaningful typographical attributes for the user to choose from. The typographical attributes displayed are: TYPEFACE, STYLE and SIZE. Note that the above attributes are not one-to-one mapped to XLFD fields of the same names.

The typographical attributes are extracted either:

- directly from the XLFD fonts announced by the underlying X11 server, *or*
- from FontSet definitions stored in the OpenWindows.fs database for the current locale. For a detailed description of the OpenWindows.fs FontSet database, please see the *XView Reference Manual*.

The second source above is used only when the font chooser operates the internationalized mode (please see description for the XtNtextFormat resource).

The FontChooser widget has the following graphical elements:

- Scrolling lists for font attributes: TYPEFACE, STYLE and SIZE.
- Numeric field for typing in the SIZE of a scalable font.
- Optional Extension container (for applications to extend the font chooser).
- Font Preview Area (can be disabled).
- Apply, Revert and Cancel buttons.

The FontChooser widget is automatically instantiated when an instance of the FontChooserShell widget is created (please see the FontChooserShell widget man page for more details).

- Components

See the *OLIT Reference Manual* for documentation.

- Coloration

See SnapShot of this feature in OpenWindows chapter.

- Core Resources

Name	**Type**	**Default**	**Access**
XtNx	Position	0	SGI
XtNy	Position	0	SGI
XtNwidth	Dimension	<Calculated>	SG
XtNheight	Dimension	<Calculated>	SG
XtNborderWidth	Dimension	0	SGI
XtNsensitive	Boolean	True	GIO
XtNancestorSensitive	Boolean	True	G
XtNscreen	Screen *	parent's	GI
XtNdepth	Cardinal	parent's	GI
XtNcolormap	Colormap	parent's	GI
XtNbackground	Pixel	parent's	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGI
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNmappedWhenMananaged	Boolean	True	SGI
XtNdestroyCallback	XtCallbackList	NULL	SGIO

- Composite Resources

See the *OLIT Reference Manual* for documentation.

- Constraint Resources

See the *OLIT Reference Manual* for documentation.

- Manager Resources

See the *OLIT Reference Manual* for documentation.

- RubberTile Resources

See the *OLIT Reference Manual* for documentation.

- FontChooser Resources

Name	Type	Default	Access
XtNapplyCallback	XtCallbackList	NULL	SGIO
XtNapplyLabel	OlStr	"Apply"	GI
XtNattributeListHeight	Dimension	<calculated>	SGI
XtNcancelCallback	XtCallbackList	NULL	SGIO
XtNcancelLabel	OlStr	"Cancel"	GI
XtNchangedCallback	XtCallbackList	NULL	SGIO
XtNcharsetInfo	String	"iso8859-1"	G
XtNerrorCallback	XtCallbackList	NULL	SGIO
XtNextensionArea	Widget	NULL	G
XtNfontSearchSpec	String	"*-*-...-* (15 *'s)	GI
XtNinitialFontName	String	NULL	GI
XtNmaximumPointSize	Cardinal	99	GI

Name	Type	Default	Access
XtNpreferredPointSizes	String	"8 10 12 14 18 24"	GI
XtNpreviewHeight	Dimension	<calculated>	SGI
XtNpreviewBorderWidth	Dimension	<calculated>	GI
XtNpreviewBackground	Pixel	XtDefaultBackground	GI
XtNpreviewFontColor	Pixel	XtDefaultForeground	GI
XtNpreviewForeground	Pixel	XtDefaultForeground	GI
XtNpreviewPresent	Boolean	True	SGI
XtNpreviewText	OlStr	"%T %S %s"	SGI
XtNrevertCallback	XtCallbackList	NULL	SGIO
XtNrevertLabel	OlStr	"Revert"	GI
XtNsizeLabel	OlStr	"Size"	GI

XtNstyleLabel	OlStr	"Style"	GI
XtNtextFormat	OlStrRep	OlGetDefaultTextFormat	GI
XtNtypefaceLabel	OlStr	"Typeface"	GI

- XtNapplyCallback

Class:	XtCCallback
Type:	XtCallbackList
Def:	NULL
Access:	SGI0
Action:	This callback is invoked when the APPLY button is selected.

The calldata structure is:

```
OlFCApplyCallbackStruct {
      intreason;
      String current_font_name;
      OlFontcurrent_font;
      }
```

Fields in the calldata structure are:

reason	OL_REASON_APPLY_FONT
current_font_name	XLFD font specification corresponding to the current choices of attributes.
current_font	XFontStruct * (XFontSet) corresponding to current_font_name when XtNtextFormat is OL_SB_STR_REP (OL_MB_STR_REP or OL_WC_STR_REP).

Note – current_font will be freed by the widget after the callback returns. Therefore, the application should use this field only for operations that are local to the callback, i.e., operations that would not depend on current_font to persist after the callback returns.

- XtNapplyLabel

Class:	XtCApplyLabel
Type:	OlStr
Def:	"Apply"
Access:	GI
Action:	This is the label displayed on the APPLY button.

Note – This resource has a localized default. This means that the default value shown is passed through `dgettext` to arrive at the final default.

- `XtNattributeListHeight`

Class:	`XtCAttributeListHeight`
Type:	`Dimension`
Def:	Calculated
Access:	SGI
Values:	`2 < XtNattributeListHeight`
Action:	This resource controls the number of items shown in the Scrolling lists. The height of the SIZE scrolling list is `XtNattributeListHeight` - 1 to accommodate the numeric field below it.

- `XtNcancelCallback`

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGI0
Action:	This callback is invoked when the CANCEL button is selected.

The calldata structure is:

```
OlFCCancelCallbackStruct {
intreason;
String current_font_name;
OlFontcurrent_font;
}
```

Fields in the calldata structure are:

`reason`	`OL_REASON_CANCEL`
`current_font_name`	XLFD font specification corresponding to the current choices of attributes.
`current_font`	`XFontStruct *` (`XFontSet`) corresponding to `current_font_name` when `XtNtextFormat` is `OL_SB_STR_REP` (`OL_MB_STR_REP` or `OL_WC_STR_REP`).

Note – `current_font` will be freed by the widget after the callback returns.

Therefore, the application should use this field only for operations that are local to the callback, i.e., operations that would not depend on `current_font` to persist after the callback returns.

- `XtNcancelLabel`

Class:	`XtCCancelLabel`
Type:	`OlStr`
Def:	"Cancel"
Access:	GI
Action:	This is the label displayed on the CANCEL button.

Note – This resource has a localized default. This means that the default value shown is passed through `dgettext` to arrive at the final default.

- `XtNchangedCallback`

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGI0
Action:	This callback is invoked when there is any change in the selections of the attributes.

The calldata structure is:

```
OlFCChangedCallbackStruct {
intreason;
String current_font_name;
OlFontcurrent_font;
String previous_font_name;
OlFontprevious_font;
}
```

Fields in the calldata structure are:

reason	OL_REASON_CHANGED_FONT
current_font_name	XLFD font specification corresponding to the current choices of attributes.
current_font	XFontStruct * (XFontSet) corresponding to the current_font_name when XtNtextFormat is OL_SB_STR_REP (OL_MB_STR_REP or OL_WC_STR_REP).
previous_font_name	XLFD font specification corresponding to the choices of attributes before the change.
previous_font	XFontStruct * (XFontSet) corresponding to previous_font_name when XtNtextFormat is OL_SB_STR_REP (OL_MB_STR_REP or OL_WC_STR_REP).

Note – current_font and previous_font will be freed by the widget after the callback returns. Therefore, the application should use this field only for operations that are local to the callback, i.e., operations that would not depend on current_font and revert_font to persist after the callback returns.

- XtNcharsetInfo

Class:	XtNCCharsetInfo
Type:	String
Def:	"iso8859-1"
Access:	G
Values:	A string containing the charset registry and encoding for fonts in this locale.
Action:	This resource is used to constrain the set of fonts used by the font chooser to a certain encoding. This is useful in locales with eight bit characters only. The resource is ignored if XtNtextFormat is not OL_SB_STR_REP and the OpenWindows.fs database is found for the locale.

Note – This resource has a localized default. This means that the default value shown is passed through dgettext to arrive at the final default.

- XtNerrorCallback

Class:	XtCCallback
Type:	XtCallbackList
Def:	NULL
Access:	SGI0
Action:	This callback is invoked when the font chooser detects an error condition. This allows the application to interpose in such cases.

The calldata structure is

```
OlFCErrorCallbackStruct {
      intreason;
      interror_num;
      String font_name;
      }
```

Fields in the calldata structure are:

`reason`	`OL_REASON_ERROR`
`error_num`	Anumber from the following table that identifies the specific case.
`current_font_name`	XLFD font specification corresponding to the current choices of attributes.

Error_num

`OL_FC_ERR_NO_FONTS_FOUND`	No fonts found
`OL_FC_ERR_NO_INITIAL_FONT`	Initial font is not matched
`OL_FC_ERR_BAD_FONT_SEARCH_SPEC`	Font Search spec is not XLFD
`OL_FC_ERR_MISSING_CHARSETS`	Incomplete font description (missing charsets)

- `XtNextensionArea`

Class:	`XtCExtensionArea`
Type:	`Widget`
Def:	`NULL`
Access:	G
Values:	A `RubberTileWidget` instantiated the first time a `tGetValues` is done on the resource.
Action:	This is the container for the application to extend the GUI of the font chooser. The extension area is a widget of class `RubberTile`. Graphically, the extension area appears above the preview area, just below the row of scrolling lists.

- XtNfontSearchSpec

Class:	XtCFontSearchSpec
Type:	String
Def:	"*-*-...-*-*"
Access:	GI
Values:	Any string that is a valid XLFD specification.
Action:	Specifies a simple regular expression that constrains the set of fonts used by the font chooser.

- XtNinitialFontName

Class:	XtCInitialFontName
Type:	String
Def:	NULL
Access:	GI
Values:	A fully specified XLFD string. If XtNtextFormat is OL_SB_STR_REP, this should be the fully-specified name of a font available on the underlying X11 server. If XtNtextFormat is not OL_SB_STR_REP, it should be the fully-specified definition of a font set in the OpenWindows.fs file for the locale.
Action:	The attributes of this font will be what is selected by the font chooser when it starts up. If initialFontName cannot be found among the fonts used by the font chooser, then the first entry in sorted order will be selected first. In addition, pressing the REVERT button anytime during an interaction with the font chooser will cause it to revert back to initialFontName.

- XtNmaximumPointSize

Class:	XtCMaximumPointSize
Type:	Cardinal
Def:	99
Access:	GI
Values:	Any positive integer
Action:	This resource is used to restrict the font chooser to operate with point-sizes up to its value.

- `XtNpreferredPointSizes`

Class:	`XtCPreferredPointSizes`
Type:	`String`
Def:	"8 10 12 14 18 24"
Access:	GI
Values:	A string containing a list of positive integers separated by space.
Action:	This resource provides the list of point sizes displayed by the font chooser in its SIZE scrolling list.

Note – This resource has a localized default. This means that the default value shown is passed through `dgettext` to arrive at the final default.

- `XtNpreviewHeight`

Class:	`XtCPreviewHeight`
Type:	`Dimension`
Def:	<calculated>
Access:	SGI
Values:	Any non-negative integer
Action:	This resource determines the height of the preview area in pixels.

- `XtNpreviewBorderWidth`

Class:	`XtCBorderWidth`
Type:	`Dimension`
Def:	<calculated>
Access:	GI
Values:	Any non-negative integer
Action:	This resource determines the border-width of the preview area in pixels.

- `XtNpreviewBackground`

Class:	`XtCBackground`
Type:	`Pixel`
Def:	`XtDefaultBackground`
Access:	GI
Values:	Any valid pixel on the screen
Action:	This resource determines the pixel value used to color the background of the preview area.

- `XtNpreviewFontColor`

Class:	`XtCFontColor`
Type:	`Pixel`
Def:	`XtDefaultForeground`
Access:	GI
Values:	Any valid pixel on the screen
Action:	This resource determines the pixel value used to color the text in the preview area.

- `XtNpreviewForeground`

Class:	`XtCForeground`
Type:	`Pixel`
Def:	`XtDefaultForeground`
Access:	GI
Values:	Any valid pixel on the screen
Action:	This resource determines the pixel value used to color the foreground in the preview area.

- `XtNpreviewPresent`

Class:	`XtCPreviewPresent`
Type:	`Boolean`
Def:	`True`
Access:	SGI
Values:	True/"true" False/"false":
Action:	This resource lets the application set whether it wants the preview area. If the application chooses not have a preview area it can set this resource to `False`.

- `XtNpreviewText`

Class:	`XtCPreviewText`
Type:	`OlStr`
Def:	"%T %S %s"
Access:	SGI
Values:	Any valid string in the widget's `textFormat`. The following three substrings are special if they appear in this string. "`%T`" This is replaced by the current typeface. "`%S`" This is replaced by the current style. "`%s`" This is replaced by the current size.
Action:	Allows the application to configure the text that the font chooser displays in its preview area.

- `XtNrevertCallback`

Class:	`XtCCallback`
Type:	`XtCallbackList`
Def:	`NULL`
Access:	SGI0
Action:	This callback is invoked when the REVERT button is selected and `XtNinitialFontName` is set to a valid XLFD specification.

The calldata structure is:

```
OlFCRevertCallbackStruct {
  int reason;
  String current_font_name;
  OlFontcurrent_font;
  Stringrevert_font_name;
  OlFontrevert_font;
}
```

Fields in the calldata structure are:

`reason`	`OL_REASON_REVERT_FONT`
`current_font_name`	XLFD font specification corresponding to the current choices of attributes.
`current_font`	`XFontStruct *` (`XFontSet`) corresponding to `current_font_name` if `XtNtextFormat` is `OL_SB_STR_REP` (`OL_MB_STR_REP` or `OL_WC_STR_REP`).
`revert_font_name`	XLFD font specification corresponding to the font that it is reverting to.
`revert_font`	`XFontStruct *` (`XFontSet`) corresponding to `revert_font_name` when `XtNtextFormat` is `OL_SB_STR_REP` (`OL_MB_STR_REP` or `OL_WC_STR_REP`).

Note – `current_font` and `revert_font` will be freed by the widget after the callback returns. Therefore, the application should use these fields only for operations that are local to the callback, i.e., operations that would not depend on `current_font` and `revert_font` to persist after the callback returns.

- `XtNrevertLabel`

Class:	`XtCRevertLabel`
Type:	`OlStr`
Def:	"Revert"
Access:	GI
Action:	This is the label displayed on the REVERT button.

Note – This resource has a localized default. This means that the default value shown is passed through `dgettext` to arrive at the final default.

- `XtNsizeLabel`

Class:	`XtCSizeLabel`
Type:	`OlStr`
Def:	"Size"
Access:	GI
Action:	This is the title of the scrolling list displaying the list of SIZE attributes.

Note – This resource has a localized default. This means that the default value shown is passed through `dgettext` to arrive at the final default.

- XtNstyleLabel

Class:	XtCStyleLabel
Type:	OlStr
Def:	"Style"
Access:	GI
Action:	This is the title of the scrolling list displaying the list of STYLE attributes.

Note – This resource has a localized default. This means that the default value shown is passed through dgettext to arrive at the final default.

- XtNtextFormat

Class:	XtCTextFormat
Type:	OlStrRep
Def:	OlDefaultTextFormat
Access:	GI
Values:	OL_SB_STR_REP, OL_MB_STR_REP, OL_WC_STR_REP
Action:	Specifies the text representation type for the widget. The toolkit computes the default setting. As the text representation type determines whether FontStruct* or XFontSet is needed for rendering text, this resource controls the source of font names for the font chooser. If the value is OL_SB_STR_REP, it uses the fonts announced by the X11 server. Otherwise, it uses fontsets specified in the locale-specific fontset database stored in: $OPENWINHOME/lib/locale/%L/OW_FONT_SETS/OpenWindows.fs.

- XtNtypefaceLabel

Class:	XtCTypefaceLabel
Type:	OlStr
Def:	"Typeface"
Access:	GI
Action:	This is the title of the scrolling list displaying the list of TYPEFACE attributes.

Note – This resource has a localized default. This means that the default value shown is passed through `dgettext` to arrive at the final default.

Activation Types

See the *OLIT Reference Manual* for documentation.

FontChooserShellWidget

Synopsis

```
#include <Xol/OpenLook.h>
#include <Xol/FontChSh.h>
```

Class

Name: `FontChooserShell`

Pointer: `fontChooserShellWidgetClass`

Ancestry:

Core -> Composite -> Shell -> WMShell -> VendorShell ->TransientShell -> FontChooserShell

Description:

The `FontChooserShell` widget provides a mechanism to directly instantiate a font chooser popup as defined in the OPEN LOOK specification.

The `FontChooserShell` internally instantiates a `FontChooser` widget as its child to provide most of the functionality. It adds the font chooser popup/down semantics defined in the OPEN LOOK specification to this basic functionality.

For details on the internals of the font chooser, please refer to its man page.

Components

See the *OLIT Reference Manual* for documentation.

Coloration

See SnapShot of this feature in OpenWindows chapter.

Core Resources

Name	Type	Default	Access
XtNx	Position	0	SGI
XtNy	Position	0	SGI
XtNwidth	Dimension	<Calculated>	SG
XtNheight	Dimension	<Calculated>	SG
XtNborderWidth	Dimension	0	SGI
XtNsensitive	Boolean	True	GIO
XtNancestorSensitive	Boolean	True	G
XtNscreen	Screen *	parent's	GI
XtNdepth	Cardinal	parent's	GI
XtNcolormap	Colormap	parent's	GI
XtNbackground	Pixel	parent's	SGID
XtNbackgroundPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNborderColor	Pixel	XtDefaultForeground	SGI
XtNborderPixmap	Pixmap	XtUnspecifiedPixmap	SGI
XtNmappedWhenMananaged	Boolean	True	SGI
XtNdestroyCallback	XtCallbackList	NULL	SGIO

Composite Resources

See the *OLIT Reference Manual* for documentation.

Shell Resources

See the *OLIT Reference Manual* for documentation.

WMShell Resources

See the *OLIT Reference Manual* for documentation.

VendorShell Resources

See the *OLIT Reference Manual* for documentation.

TransientShell Resources

See the *OLIT Reference Manual* for documentation.

FontChooserShell Resources

Name	Type	Default	Access
XtNfontChooserWidget	Widget	<determined>	G
XtNtextFormat	OlStrRep	OlGetDefaultTextFormat	GI

- XtNfontChooserWidget

Class:	XtCWidget
Type:	Widget
Access:	G
Values:	contains the FontChooser child widget that can be accessed for setting/ getting its resources (please see FontChooser widget man page for its resources).

- XtNtextFormat

Class:	XtCTextFormat
Type:	OlStrRep
Def:	OlDefaultTextFormat
Access:	GI
Values:	OL_SB_STR_REP, OL_MB_STR_REP, OL_WC_STR_REP
Action:	Its value is propagated to the child FontChooser widget.

Activation Types

See the *OLIT Reference Manual* for documentation.

Sample CG3 Source with Sun Interface

This section of the appendix shows you the changes you have to make to the CG3 framebuffer to add support for the Sun interface. This section first shows the sample CG3 DDX handler from MIT, and then shows the CG3 DDX handler for OpenWindows Version 4. With very few changes, the CG3 DDX handler supports Sun-specific interfaces.

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 1 of 7)

```
/*-
 * sunCG3C.c --
 *  Functions to support the sun CG3 board as a memory frame buffer.
 */
```

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 2 of 7)

```
/************************************************************
Copyright 1987 by Sun Microsystems, Inc. Mountain View, CA.

                    All Rights Reserved

Permission  to  use,  copy,  modify,  and  distribute   this
software  and  its documentation for any purpose and without
fee is hereby granted, provided that the above copyright no-
tice  appear  in all copies and that both that copyright no-
tice and this permission notice appear in  supporting  docu-
mentation,  and  that the names of Sun or MIT not be used in
advertising or publicity pertaining to distribution  of  the
software  without specific prior written permission. Sun and
M.I.T. make no representations about the suitability of this
software for any purpose. It is provided "as is" without any
express or implied warranty.

SUN DISCLAIMS ALL WARRANTIES WITH REGARD TO  THIS  SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FIT-
NESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SUN BE  LI-
ABLE  FOR  ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR
ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,  DATA  OR
PROFITS,  WHETHER  IN  AN  ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION  WITH
THE USE OR PERFORMANCE OF THIS SOFTWARE.

********************************************************/

#include    "sun.h"

#include    <sys/mman.h>
#include    <pixrect/memreg.h>
/*
#include    <sundev/cg4reg.h>
*/
#include    "colormap.h"
#include    "colormapst.h"
#include    "resource.h"
#include    <struct.h>

/*-
 * The cg3 frame buffer is divided into several pieces.
 *  1) an array of 8-bit pixels
 *  2) a one-bit deep overlay plane
 *  3) an enable plane
```

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 3 of 7)

```
 *  4) a colormap and status register
 *
 * XXX - put the cursor in the overlay plane
 */

#define CG3A_HEIGHT      900
#define CG3A_WIDTH       1152
#define CG3B_HEIGHT 768
#define CG3B_WIDTH 1024

typedef struct cg3ac {
#ifdef sparc
    u_char mpixel[128*1024];/* bit-per-pixel memory */
    u_char epixel[128*1024];/* enable plane */
#endif
        u_char cpixel[CG3A_HEIGHT][CG3A_WIDTH];   /* byte-per-pixel memory */
} CG3AC, CG3ACRec, *CG3ACPtr;

typedef struct cg3bc {
#ifdef sparc
    u_char mpixel[128*1024];/* bit-per-pixel memory */
    u_char epixel[128*1024];/* enable plane */
#endif
        u_char cpixel[CG3B_HEIGHT][CG3B_WIDTH];   /* byte-per-pixel memory */
} CG3BC, CG3BCRec, *CG3BCPtr;

#define CG3AC_IMAGE(fb)      ((caddr_t)((fb)->cpixel))
#define CG3AC_IMAGELEN       (((sizeof (CG3AC) + 4095)/4096)*4096)
#define CG3BC_IMAGE(fb)      ((caddr_t)((fb)->cpixel))
#define CG3BC_IMAGELEN       (((sizeof (CG3BC) + 4095)/4096)*4096)

/*-
 *-----------------------------------------------------------------------
 * sunCG3CInit --
 *  Attempt to find and initialize a cg3 framebuffer
 *
 * Results:
 *  TRUE if everything went ok. FALSE if not.
 *
 * Side Effects:
 *  Most of the elements of the ScreenRec are filled in. Memory is
 *  allocated for the frame buffer and the buffer is mapped. The
 *  video is enabled for the frame buffer...
 *
 *-----------------------------------------------------------------------
```

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 4 of 7)

```
 */
/*ARGSUSED*/
static Bool
sunCG3CInit (index, pScreen, argc, argv)
    int      index;    /* The index of pScreen in the ScreenInfo */
    ScreenPtr  pScreen;  /* The Screen to initialize */
    int      argc;    /* The number of the Server's arguments. */
    char      **argv;   /* The arguments themselves. Don't change! */
{
    if (!cfbScreenInit (pScreen, sunFbs[index].fb,
           sunFbs[index].info.fb_width,
           sunFbs[index].info.fb_height,
           monitorResolution, monitorResolution,
           sunFbs[index].info.fb_width))
    return (FALSE);

    if (!sunScreenAllocate (pScreen))
    return FALSE;

    sunCGScreenInit (pScreen);

    if (!sunScreenInit (pScreen))
    return FALSE;

    sunSaveScreen (pScreen, SCREEN_SAVER_OFF);

    return cfbCreateDefColormap(pScreen);
}

/*-
 *-----------------------------------------------------------------------
 * sunCG3CProbe --
 *  Attempt to find and initialize a cg3 framebuffer
 *
 * Results:
 *  TRUE if everything went ok. FALSE if not.
 *
 * Side Effects:
 *  Memory is allocated for the frame buffer and the buffer is mapped.
 *
 *-----------------------------------------------------------------------
 */
/*ARGSUSED*/
Bool
sunCG3CProbe (pScreenInfo, index, fbNum, argc, argv)
```

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 5 of 7)

```
    ScreenInfo  *pScreenInfo;/* The screenInfo struct */
    int       index;    /* The index of pScreen in the ScreenInfo */
    int       fbNum;    /* Index into the sunFbData array */
    int       argc;    /* The number of the Server's arguments. */
    char       **argv;   /* The arguments themselves. Don't change! */
{
    int         fd;
    struct fbtype fbType;
    pointerfb;
    CG3ACPtr CG3ACfb;
    CG3BCPtr CG3BCfb;

    if ((fd = sunOpenFrameBuffer(FBTYPE_SUN3COLOR, &fbType, index, fbNum,
                argc, argv)) < 0)
    return FALSE;

#ifdef_MAP_NEW
    if (fbType.fb_width == CG3A_WIDTH) {
    if ((int)(CG3ACfb = (CG3ACPtr) mmap((caddr_t) 0,
         CG3AC_IMAGELEN,
         PROT_READ | PROT_WRITE,
         MAP_SHARED | _MAP_NEW, fd, 0)) == -1) {
        Error("Mapping cg3c");
        (void) close(fd);
        return FALSE;
    }
    fb = (pointer) CG3AC_IMAGE(CG3ACfb);
    }
    else if (fbType.fb_width == CG3B_WIDTH) {
    if ((int)(CG3BCfb = (CG3BCPtr) mmap((caddr_t) 0,
         CG3BC_IMAGELEN,
         PROT_READ | PROT_WRITE,
         MAP_SHARED | _MAP_NEW, fd, 0)) == -1) {
        Error("Mapping cg3c");
        (void) close(fd);
        return FALSE;
    }
    fb = (pointer) CG3BC_IMAGE(CG3BCfb);
    }
    else {
        Error("Mapping cg3c");
        (void) close(fd);
        return FALSE;
    }
#else_MAP_NEW
```

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 6 of 7)

```
     if (fbType.fb_width == CG3A_WIDTH) {
    CG3ACfb = (CG3ACPtr) valloc(CG3AC_MONOLEN +
        CG3AC_ENBLEN + CG3AC_IMAGELEN);
    if (CG3ACfb == (CG3ACPtr) NULL) {
        ErrorF("Could not allocate room for frame buffer.\n");
        return FALSE;
    }

    if (mmap((caddr_t) CG3ACfb, CG3AC_MONOLEN +
        CG3AC_ENBLEN + CG3AC_IMAGELEN,
        PROT_READ | PROT_WRITE,
        MAP_SHARED, fd, 0) < 0) {
        Error("Mapping cg3c");
        (void) close(fd);
        return FALSE;
    }
    fb = (pointer) CG3AC_IMAGE(CG3ACfb);
     }
     else if (fbType.fb_width == CG3B_WIDTH) {
    CG3BCfb = (CG3BCPtr) valloc(CG3BC_MONOLEN +
        CG3BC_ENBLEN + CG3BC_IMAGELEN);
    if (CG3BCfb == (CG3BCPtr) NULL) {
        ErrorF("Could not allocate room for frame buffer.\n");
        return FALSE;
    }

    if (mmap((caddr_t) CG3BCfb, CG3BC_MONOLEN +
        CG3BC_ENBLEN + CG3BC_IMAGELEN,
        PROT_READ | PROT_WRITE,
        MAP_SHARED, fd, 0) < 0) {
        Error("Mapping cg3c");
        (void) close(fd);
        return FALSE;
    }
    fb = (pointer) CG3BC_IMAGE(CG3BCfb);
     }
     else {
        Error("Mapping cg3c");
        (void) close(fd);
        return FALSE;
     }
#endif_MAP_NEW

     sunFbs[index].fd = fd;
     sunFbs[index].info = fbType;
```

Code Example 16-10 Sample CG3 Driver From MIT (Sheet 7 of 7)

```
    sunFbs[index].fb = fb;
    sunSupportsDepth8 = TRUE;
    return TRUE;
}

/*ARGSUSED*/
Bool
sunCG3CCreate(pScreenInfo, argc, argv)
    ScreenInfo  *pScreenInfo;
    int      argc;
    char      **argv;
{
    return (AddScreen(sunCG3CInit, argc, argv) >= 0);
}
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 1 of 7)

```
/* @(#)sunCG3C.c1.13 13 Jan 1993 */
/*-
 * sunCG3C.c --
 *  Functions to support the sun CG3 board as a memory frame buffer.
 */

/****************************************************************/
/* Modified for loadable modules by Milind Pansare*/
/****************************************************************/
/************************************************************
Copyright 1987 by Sun Microsystems, Inc. Mountain View, CA.

                    All Rights Reserved

Permission  to  use,  copy,  modify,  and  distribute   this
software  and  its documentation for any purpose and without
fee is hereby granted, provided that the above copyright no-
tice  appear  in all copies and that both that copyright no-
tice and this permission notice appear in  supporting  docu-
mentation,  and  that the names of Sun or MIT not be used in
advertising or publicity pertaining to distribution  of  the
software  without specific prior written permission. Sun and
M.I.T. make no representations about the suitability of this
software for any purpose. It is provided "as is" without any
express or implied warranty.

SUN DISCLAIMS ALL WARRANTIES WITH REGARD TO  THIS  SOFTWARE,
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 2 of 7)

```
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FIT-
NESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SUN BE  LI-
ABLE  FOR  ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR
ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,  DATA  OR
PROFITS,  WHETHER  IN  AN  ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION  WITH
THE USE OR PERFORMANCE OF THIS SOFTWARE.

********************************************************/

#include    "sun.h"
#include <servermd.h>

#include    <sys/mman.h>
#ifdef SVR4
#include    <sys/memreg.h>
#else
#include    <pixrect/memreg.h>
#include    <struct.h>
#endif
#include    "colormap.h"
#include    "colormapst.h"
#include    "resource.h"

#if defined(__STDC__)
static const PixmapFormatRecformats[] = {
#else /* !__STDC__ */
static PixmapFormatRecformats[] = {
#endif /* __STDC__ */
    1,  1,  BITMAP_SCANLINE_PAD,
    8,  8,  BITMAP_SCANLINE_PAD,
};
#ifdef NUMFORMATS
#undef NUMFORMATS
#endif /* NUMFORMATS */
#define NUMFORMATS(sizeof formats)/(sizeof formats[0])

/*-
 *-----------------------------------------------------------------------
 * sunCG3Init --
 *  Attempt to find and initialize a cg3 framebuffer
 *
 * Results:
 *  TRUE if everything went ok. FALSE if not.
 *
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 3 of 7)

```
 * Side Effects:
 *  Most of the elements of the ScreenRec are filled in. Memory is
 *  allocated for the frame buffer and the buffer is mapped. The
 *  video is enabled for the frame buffer...
 *
 *-----------------------------------------------------------------------
 */
/*ARGSUSED*/
Bool
sunCG3Init (index, pScreen, argc, argv)
    int       index;    /* The index of pScreen in the ScreenInfo */
    ScreenPtr  pScreen;  /* The Screen to initialize */
    int       argc;    /* The number of the Server's arguments. */
    char       **argv;   /* The arguments themselves. Don't change! */
{
    sunScreenPtr pPrivate;
    int i;
    int     dpix, dpiy;
    int     defClass, defDepth;
    Bool    grayVis;
    extern Bool sunCG3CloseScreen();

    if (!sunScreenAllocate (pScreen))
    return FALSE;
    pPrivate = GetScreenPrivate(pScreen);

    if (!sunCG3Probe(index, pScreen, argc, argv)) {
    return FALSE;
    }

    /* Advertise the PixmapFormats that this screen can support */
    for (i=0; i< NUMFORMATS; i++) {
    if (!sunSetPixmapFormat(&formats[i])) {
        return FALSE;
    }
    }

    sunGetMonitorRes(index, &dpix, &dpiy);
    sunGetVisualInfo(index, &defClass, &defDepth, &grayVis);

    /*
     * Check for bad defDepth.  Since cg6 supports ALL_VISUALS we don't
     * need to check defClass.
     */
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 4 of 7)

```
    if (defDepth && (defDepth != 8)) {
        ErrorF("cgsix only supports depth 8 visual classes\n");
        defDepth = 8;
    }

    cfbSetDefaultVisual(defClass, defDepth);

    /*
     * Advertise the depths & visuals for each depth that cfb should create
     * cfbSetVisualTypes(depth, visualsToExport, bitsPerRGBValue)
    */
   cfbSetVisualTypes(1, NO_VISUALS, 1); /* depth 1 pixmaps are always supported
*/
    if (grayVis)
        cfbSetVisualTypes(8, GRAY_VISUALS, 8);
    else
        cfbSetVisualTypes(8, ALL_VISUALS, 8);

    if (!cfbScreenInit (pScreen, pPrivate->sunFbs.fb,
           pPrivate->sunFbs.info.fb_width,
           pPrivate->sunFbs.info.fb_height,
           dpix, dpiy,
           pPrivate->sunFbs.info.fb_width))
    return (FALSE);

    sunCGScreenInit (pScreen);

    if (!sunScreenInit (pScreen))
    return FALSE;

    /* Wrap pScreen->CloseScreen */
    pPrivate->CloseScreen = pScreen->CloseScreen;
    pScreen->CloseScreen = sunCG3CloseScreen;

    sunSaveScreen (pScreen, SCREEN_SAVER_OFF);

    return cfbCreateDefColormap(pScreen);
}

/*-
 *-----------------------------------------------------------------------
 * sunCG3Probe --
 *  Map the cg3 framebuffer
 *
 * Results:
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 5 of 7)

```
 *  TRUE if everything went ok. FALSE if not.
 *
 * Side Effects:
 *  Memory is allocated for the frame buffer and the buffer is mapped.
 *
 *-----------------------------------------------------------------------
 */
/*ARGSUSED*/
static Bool
sunCG3Probe(index, pScreen, argc, argv)
    int       index;    /* The index of pScreen in the ScreenInfo */
    ScreenPtr  pScreen;/* Pointer to Screen structure */
    int       argc;    /* The number of the Server's arguments. */
    char       **argv;   /* The arguments themselves. Don't change! */
{
    int          fd;
    struct fbtype fbType;
    int pagemask, mapsize;
    int imagelen;
    caddr_tmapaddr;
    caddr_taddr;
    off_t offset;
    static fbFd sunFbs;
    SetupScreen(pScreen);

    if ((fd = sunOpenFrameBuffer(FBTYPE_SUN3COLOR, &fbType, index,
    &pPrivate->sunFbs, argc, argv)) < 0)
    return FALSE;

#ifdef SVR4
    pagemask = sysconf(_SC_PAGESIZE) - 1;
#else
    pagemask = getpagesize() - 1;
#endif
    imagelen = ((fbType.fb_width + 1) & ~1) * fbType.fb_height;
    mapsize = (imagelen + pagemask) & ~pagemask;
    offset = (((((((fbType.fb_width + 15) >> 3) & ~1) * fbType.fb_height)
         + pagemask) & ~pagemask) << 1); /* Take a deep breath */
    addr = 0;

#ifndef_MAP_NEW
    addr = (caddr_t) valloc(mapsize);
    if (addr == (caddr_t) NULL) {
    ErrorF("Could not allocate room for frame buffer.\n");
    (void) close (fd);
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 6 of 7)

```
    return FALSE;
    }
#endif_MAP_NEW

    mapaddr = (caddr_t) mmap((caddr_t) addr,
         mapsize,
         PROT_READ | PROT_WRITE,
         MAP_SHARED, fd, offset);

    if (mapaddr == (caddr_t) -1) {
    Error("Mapping cg3");
    (void) close(fd);
    return FALSE;
    }

    if (mapaddr == 0)
        mapaddr = addr;

    pPrivate->sunFbs.fd = fd;
    strcpy(pPrivate->sunFbs.devName, GetDevname(index));
    pPrivate->sunFbs.info = fbType;
    pPrivate->sunFbs.fb = (pointer)mapaddr;
    pPrivate->sunFbs.maplength = mapsize;
    pPrivate->sunFbs.EnterLeave = NULL;
    return TRUE;
}

/*-
 *-----------------------------------------------------------------------
 * sunCG3CloseScreen
 *  Close the cg3 device. Closes the file descriptor, and munmaps itself.
 *
 * Results:
 *  TRUE if everything went ok. FALSE if not.
 *
 * Side Effects:
 *  Closes the file descriptor, closes the memory mapping.
 *
 *-----------------------------------------------------------------------
 */
static Bool
sunCG3CloseScreen(i, pScreen)
    int i;
    ScreenPtrpScreen;
{
```

Code Example 16-11 Sample CG3 Driver Written To Support Sun Interface (Sheet 7 of 7)

```
    SetupScreen(pScreen);
    Bool    ret;

    sunDisableCursor (pScreen);
    (void) (*pScreen->SaveScreen) (pScreen, SCREEN_SAVER_OFF);
    pScreen->CloseScreen = pPrivate->CloseScreen;
    ret = (*pScreen->CloseScreen) (i, pScreen);
    if (munmap((caddr_t)(pPrivate->sunFbs.fb),
          (size_t)(pPrivate->sunFbs.maplength)) == -1) {
    Error("Unmapping cg3");
    }
    (void) close(pPrivate->sunFbs.fd);
    xfree ((pointer) pPrivate);
    return ret;
}
```

Sample Tablet Device Driver Using the X Input Extension

The following is sample source for writing a tablet device using the X Input Extension. The code is in two parts. The first part is the source file that handles X input. This file is compiled into dynamic shared libraries that the server loads when a client requests this type of input device. Second, there is a source file for the STREAMS module that listens to the input device and sends a stream-based VUID to the input extension.

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 1 of 13)

```
/*-
 * sunTablet.c --
 *  Functions for playing with the tablet device.
 *
 * Copyright (c) 1987 by the Regents of the University of California
 *
 * Permission to use, copy, modify, and distribute this
 * software and its documentation for any purpose and without
 * fee is hereby granted, provided that the above copyright
 * notice appear in all copies.  The University of California
 * makes no representations about the suitability of this
 * software for any purpose.  It is provided "as is" without
 * express or implied warranty.
 *
 *
 */

/*************************************************************
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 2 of 13)

```
Copyright 1987 by Sun Microsystems, Inc. Mountain View, CA.

                    All Rights Reserved

Permission  to  use,  copy,  modify,  and  distribute   this
software  and  its documentation for any purpose and without
fee is hereby granted, provided that the above copyright no-
tice  appear  in all copies and that both that copyright no-
tice and this permission notice appear in  supporting  docu-
mentation,  and  that the names of Sun or MIT not be used in
advertising or publicity pertaining to distribution  of  the
software  without specific prior written permission. Sun and
M.I.T. make no representations about the suitability of this
software for any purpose. It is provided "as is" without any
express or implied warranty.

SUN DISCLAIMS ALL WARRANTIES WITH REGARD TO  THIS  SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FIT-
NESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SUN BE  LI-
ABLE  FOR  ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR
ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,  DATA  OR
PROFITS,  WHETHER  IN  AN  ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION  WITH
THE USE OR PERFORMANCE OF THIS SOFTWARE.

********************************************************/

#ifndeflint
static char sccsid[] = "%W %G Copyright 1987 Sun Micro";
#endif

#define NEED_EVENTS
#include    "sun.h"
#include    "mipointer.h"
#include    "XI.h"
#include    "XInput.h"
#include    "XIproto.h"
#include    "/home/dhuff/mouse/tablet/tbltio.h"
#include <fcntl.h>
#include <sys/filio.h>
#ifdef SVR4
#include <signal.h>
#include <sys/stropts.h>
#endif
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 3 of 13)

```
#define TBLT_RES 20000

typedef struct {
    int fd;
    int         mux_fd;         /* Descriptor to mux device */
    int         index;          /* kbd index once AddEnabledDevice has
                * been called */
    int bmask;/* current button state */
    Atom    sun_tablet_atom;
    int x, y;
    unsigned char xAxis, yAxis;
    BoolcorePointer;
} sunTabPrivRec, *sunTabPrivPtr;


XI_eventPtr sunTabletGetEvents();
void   sunTabletEnqueueEvent();
extern int              DeviceMotionNotify;
extern int              DeviceValuator;
extern int              DeviceButtonPress;
extern int              DeviceButtonRelease;
extern int              ProximityIn;
extern int              ProximityOut;

static voidsunTabletCtrl();


/*-
 *-----------------------------------------------------------------------
 * sunTabletProc --
 *  Handle the initialization, etc. of the tablet device.
 *
 * Results:
 *  none.
 *
 * Side Effects:
 *
 * Note:
 *  When using sunwindows, all input comes off a single fd, stored in the
 *  global windowFd.  Therefore, only one device should be enabled and
 *  disabled, even though the application still sees both tablet and
 *  keyboard.  We have arbitrarily chosen to enable and disable windowFd
 *  in the keyboard routine sunKbdProc rather than in sunMouseProc.
 *
 *-----------------------------------------------------------------------
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 4 of 13)

```
 */
int
sunTabletProc (DevicePtr pTablet, int what)
{
    register int  fd;
    int       format, i;
    static int  oformat;
    char  *device, *getenv ();
    extern int    sunTabletMotionProc();
    extern int  sunTabletAxisChange();
    extern int  sunTabletSetValuators();
    BYTE  map[4];
    sunTabPrivPtr  pPriv;

    switch (what) {
    case DEVICE_INIT:
        if (pTablet->devicePrivate)
         return(!Success);/* DEVICE_INIT should only be
                    * called once */
        pPriv = (sunTabPrivPtr)Xalloc(sizeof(sunTabPrivRec));
        fd = open ("/dev/ttya", O_RDWR, 0);
        if (fd < 0) {
            Error ("Opening /dev/ttya");
            return (!Success);
        }
            if (ioctl(fd, I_POP, 0) < 0) {
                Error("Error popping ttya");
            return (!Success);
            }
            if (ioctl(fd, I_POP, 0) < 0) {
                Error("Error popping ttya");
            return (!Success);
            }
        if (fcntl (fd, F_SETFL, (FNDELAY|FASYNC)) < 0) {
            perror("sunTabletProc");
            ErrorF("Can't set up tablet on fd %d\n", fd);
        }
        if (ioctl(fd, I_PUSH, "tblt") < 0) {
            perror("sunTabletProc");
            ErrorF("ioctl I_PUSH failed");
            }
        pPriv->fd = fd;
        pPriv->bmask = 0;
        pPriv->x = 0;
        pPriv->y = 0;
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 5 of 13)

```
        pPriv->xAxis = 0;
        pPriv->yAxis = 1;
        pPriv->corePointer = FALSE;
       pPriv->sun_tablet_atom = MakeAtom(XI_TABLET, strlen(XI_TABLET), FALSE);
        pTablet->devicePrivate = (pointer) pPriv;

        pTablet->on = FALSE;
        map[1] = 1;
        map[2] = 2;
        map[3] = 3;

        InitButtonClassDeviceStruct(pTablet, 3, map);
            InitValuatorClassDeviceStruct(pTablet, 3, sunTabletMotionProc,
                                 0, Relative);
        InitPtrFeedbackClassDeviceStruct(pTablet, sunTabletCtrl);

        for (i=0; i<3; i++)
            InitValuatorAxisStruct(pTablet, i, 0, 0, TBLT_RES,
                              TBLT_RES, TBLT_RES);
        InitFocusClassDeviceStruct(pTablet);
        InitProximityClassDeviceStruct(pTablet);

            AssignTypeAndName(pTablet, pPriv->sun_tablet_atom, XI_TABLET);
        RegisterXPointerInterest(pTablet, TRUE, sunTabletAxisChange);
        RegisterSetDeviceValuatorsProc(pTablet, sunTabletSetValuators);
            RegisterFdIo(pTablet, pPriv->fd,
                 sunTabletGetEvents, sunTabletEnqueueEvent);

        break;

    case DEVICE_ON:
        pPriv = (sunTabPrivPtr)(pTablet->devicePrivate);
            if (ioctl(pPriv->fd, VUIDGFORMAT, &oformat) < 0) {
         Error("sunTabletProc VUIDGFORMAT");
         return(!Success);
            }
        format = VUID_FIRM_EVENT;
            if (ioctl(pPriv->fd, VUIDSFORMAT,
               &format) < 0) {
         Error("sunTabletProc VUIDSFORMAT");
         return(!Success);
            }
        ioctl(pPriv->fd, I_SETSIG, S_INPUT);
        AddEnabledDevice (pPriv->fd, &pPriv->mux_fd, &pPriv->index);
        pTablet->on = TRUE;
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 6 of 13)

```
        break;

    case DEVICE_OFF:

        RemoveEnabledDevice (((sunTabPrivPtr)pTablet->devicePrivate)->fd);
        pTablet->on = FALSE;
        break;

    case DEVICE_CLOSE:

        pPriv = (sunTabPrivPtr)(pTablet->devicePrivate);
        if (pTablet->on)
            RemoveEnabledDevice (pPriv->fd);
        if (ioctl (pPriv->fd, VUIDSFORMAT, &oformat) < 0)
              Error("VUIDSFORMAT");
        close(pPriv->fd);
        xfree(pTablet->devicePrivate);
        pTablet->on = FALSE;
        break;
    }
    return (Success);
}

/*-
 *-----------------------------------------------------------------------
 * sunTabletCtrl --
 *  Alter the control parameters for the tablet. Since acceleration
 *  etc. is done from the PtrCtrl record in the tablet's device record,
 *  there's nothing to do here.
 *
 * Results:
 *  None.
 *
 * Side Effects:
 *  None.
 *
 *-----------------------------------------------------------------------
 */
/*ARGSUSED*/
static void
sunTabletCtrl (DevicePtr pTablet)
{
}

/*-
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 7 of 13)

```
 *----------------------------------------------------------------------
 * sunTabletGetEvents --
 *  Return the events waiting in the wings for the given tablet.
 *
 * Results:
 *  A pointer to an array of Firm_events or (Firm_event *)0 if no events
 *  The number of events contained in the array.
 *  A boolean as to whether more events might be available.
 *
 * Side Effects:
 *  None.
 *----------------------------------------------------------------------
 */
static XI_eventPtr
sunTabletGetEvents (DevicePtr pTablet, int *pNumEvents, Bool *pAgain)
{
    int      i, nBytes;    /* number of bytes of events available. */
    register sunTabPrivPtr  pPriv;
    static Firm_event evBuf[MAXEVENTS];
    static XI_event tabletBuf[MAXEVENTS];
    struct timeval      now;

    pPriv = (sunTabPrivPtr) pTablet->devicePrivate;

    if(ioctl(pPriv->fd, FIONREAD, &nBytes) == -1 || nBytes == 0) {
      *pNumEvents = 0;
      *pAgain = FALSE;
      return( (XI_eventPtr)NULL );
    }

    nBytes = read (pPriv->fd, (char *)evBuf, MAXEVENTS*sizeof(Firm_event));

    if (nBytes < 0) {
    if (errno == EWOULDBLOCK) {
        *pNumEvents = 0;
        *pAgain = FALSE;
        } else {
        Error("Reading tablet");
        FatalError("Could not read from tablet");
        }
    } else {
    *pNumEvents = nBytes / sizeof(Firm_event);
    *pAgain = (nBytes == sizeof(evBuf));
    }
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 8 of 13)

```
    for (i=0; i<(nBytes/sizeof(Firm_event)); i++) {
              tabletBuf[i].time = evBuf[i].time;
              tabletBuf[i].event = (pointer) &(evBuf[i]);
    }

    return (tabletBuf);
}

/*-
 *-----------------------------------------------------------------------
 * TabletAccelerate --
 *      Given a delta and a tablet, return the acceleration of the delta.
 *
 * Results:
 *      The corrected delta
 *
 * Side Effects:
 *      None.
 *
 *-----------------------------------------------------------------------
 */
static short
TabletAccelerate (DevicePtr pTablet, int delta)
{
    register int  sgn = sign(delta);
    register PtrCtrl *pCtrl;

    delta = abs(delta);
    pCtrl = &((DeviceIntPtr) pTablet)->ptrfeed->ctrl;

    if (delta > pCtrl->threshold) {
        return ((short) (sgn * (pCtrl->threshold +
                                ((delta - pCtrl->threshold) * pCtrl->num) /
                                pCtrl->den)));
    } else {
        return ((short) (sgn * delta));
    }
}

/*-
 *-----------------------------------------------------------------------
 * sunTabletEnqueueEvent --
 *  Given a Firm_event for a tablet, pass it off the the dix layer
 *  properly converted...
 *
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 9 of 13)

```
 * Results:
 *  None.
 *
 * Side Effects:
 *  The cursor may be redrawn...? devPrivate/x/y will be altered.
 *
 *-----------------------------------------------------------------------
 */
static void
sunTabletEnqueueEvent (DevicePtr pTablet, XI_eventPtr Xev)
{
    register sunTabPrivPtrpPriv;/* Private data for pointer */
    Firm_event *ev = (Firm_event *) Xev->event;
    int x, y;
    static intpressure = 0;
    register unsigned long time;
    register intbmask;/* Temporary button mask */

    pPriv = (sunTabPrivPtr)pTablet->devicePrivate;

    if (pPriv->corePointer) {
    xEvent  xE;

    time = xE.u.keyButtonPointer.time = TVTOMILLI(ev->time);
    switch(ev->id)
    {
    case TBLT_BUTTON_1:
    case TBLT_BUTTON_2:
    case TBLT_BUTTON_3:
    case TBLT_BUTTON_4:
        xE.u.u.detail = (ev->id - TBLT_BUTTON_1) + 1;
        bmask = 1 << xE.u.u.detail;
        if (ev->value == VKEY_UP) {
        if (pPriv->bmask & bmask) {
            xE.u.u.type = ButtonRelease;
            pPriv->bmask &= ~bmask;
                } else {
            return;
                }
            } else {
        if ((pPriv->bmask & bmask) == 0) {
            xE.u.u.type = ButtonPress;
            pPriv->bmask |= bmask;
                } else {
            return;
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 10 of 13)

```
                    }
                }
            mieqEnqueue(&xE);
            break;
            case TBLT_PROXIMITY:
            case TBLT_PRESSURE:
            break;
        case TBLT_X_DELTA:
            if (pPriv->xAxis == 0)
                miPointerDeltaCursor(TabletAccelerate(pTablet,ev->value), 0,
                        time);
                else
                miPointerDeltaCursor(0, TabletAccelerate(pTablet,ev->value),
                        time);
            break;
            case TBLT_Y_DELTA:
            if (pPriv->yAxis == 1)
                miPointerDeltaCursor(0, -TabletAccelerate(pTablet,ev->value),
                        time);
                else
                miPointerDeltaCursor(-TabletAccelerate(pTablet,ev->value), 0,
                        time);
                break;
            case TBLT_X_ABSOLUTE:
            miPointerPosition(&x, &y);
            if (pPriv->xAxis == 0)
                miPointerAbsoluteCursor(ev->value, y, time);
                else
                miPointerAbsoluteCursor(x, ev->value, time);
            break;
            case TBLT_Y_ABSOLUTE:
            miPointerPosition(&x, &y);
            if (pPriv->yAxis == 1)
                miPointerAbsoluteCursor(x, ev->value, time);
                else
                miPointerAbsoluteCursor(ev->value, y, time);
            break;
            default:
            FatalError("sunTabletEnqueueEvent: unrecognized id\n");
            break;
            }
        } else {
            xEvent xE[2];
            deviceKeyButtonPointer *xev = (deviceKeyButtonPointer *) xE;
            deviceValuator *xv;
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 11 of 13)

```
        xev->time = TVTOMILLI(ev->time);
        miPointerPosition(&x, &y);
        xev->root_x = x;
        xev->root_y = y;

        switch(ev->id)
        {
        case TBLT_BUTTON_1:
        case TBLT_BUTTON_2:
        case TBLT_BUTTON_3:
            xev->deviceid = (int) (((DeviceIntPtr)pTablet)->id);
            xev->detail = (ev->id - TBLT_BUTTON_1) + 1;
        bmask = 1 << xev->detail;
        if (ev->value == VKEY_UP) {
        if (pPriv->bmask & bmask) {
            xev->type = DeviceButtonRelease;
            pPriv->bmask &= ~bmask;
                } else {
            return;
                }
            } else {
        if ((pPriv->bmask & bmask) == 0) {
            xev->type = DeviceButtonPress;
            pPriv->bmask |= bmask;
                } else {
            return;
                }
            }
            mieqEnqueue(&xE[0]);
            return;
        case TBLT_X_DELTA:
            pPriv->x += ev->value;
            xev->type = DeviceMotionNotify;
            break;
        case TBLT_Y_DELTA:
            pPriv->y -= ev->value;
            xev->type = DeviceMotionNotify;
            break;
        case TBLT_X_ABSOLUTE:
            pPriv->x = ev->value;
            xev->type = DeviceMotionNotify;
            break;
        case TBLT_Y_ABSOLUTE:
            pPriv->y = ev->value;
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 12 of 13)

```
            xev->type = DeviceMotionNotify;
            break;
        case TBLT_PRESSURE:
            pressure = ev->value;
            xev->type = DeviceMotionNotify;
            break;
        case TBLT_PROXIMITY:
            if (ev->value == 0)
                xev->type = ProximityOut;
            else
                xev->type = ProximityIn;
            break;
        default:
            FatalError ("sunTabletEnqueueEvent: unrecognized id\n");
            break;
        }
        xev->deviceid = (int) (((DeviceIntPtr)pTablet)->id) | MORE_EVENTS;
        xev->detail = 0;

        xv = (deviceValuator *) ++xev;
        xv->type = DeviceValuator;
        xv->deviceid = (int) (((DeviceIntPtr)pTablet)->id);
        xv->num_valuators = 3;
        xv->device_state = 0;
        xv->first_valuator = 0;
        xv->valuator0 = pPriv->x;
        xv->valuator1 = pPriv->y;
        xv->valuator2 = pressure;

        mieqEnqueue(&xE[0]);
        mieqEnqueue(&xE[1]);
    }
}

static int
sunTabletMotionProc(DevicePtr pPtr, xTimecoord *coords, unsigned long start,
            unsigned long stop, ScreenPtr pScreen)
{
   return(0);
}

static int
sunTabletAxisChange(DevicePtr pTablet, Bool isCore, unsigned char x,
            unsigned char y)
{
```

Code Example 16-12 Sample Source For X Input Extension For Tablet Device (Sheet 13 of 13)

```
    sunTabPrivPtrpPriv;/* Private data for pointer */

    pPriv = (sunTabPrivPtr)pTablet->devicePrivate;

    if (isCore) {
    if ((x > 1) || (y > 1) || (x == y))
        return (!Success);

    pPriv->xAxis = x;
    pPriv->yAxis = y;
    pPriv->corePointer = TRUE;
    } else
    pPriv->corePointer = FALSE;

    return (Success);
}

static int
sunTabletSetValuators(DevicePtr pTablet, int *valuators, int first_valuator,
              int num_valuators)
{
    sunTabPrivPtrpPriv;/* Private data for pointer */

    pPriv = (sunTabPrivPtr)pTablet->devicePrivate;

    /* Only allow the setting of the x and y axis (0 and 1)*/

    if (first_valuator + num_valuators > 2)
    return (!Success);

    while (num_valuators--) {
        if (first_valuator == 0) {
        if (pPriv->xAxis == 0)
            pPriv->x = *valuators;
            else
            pPriv->y = *valuators;
            valuators++;
        first_valuator++;
        mieqUpdateKbdPtr(NULL, pTablet);
        }
    }
    return (Success);
}
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 1 of 36)

```
/*
 * Copyright (c) 1987 by Sun Microsystems, Inc.
 */

#ident"@(#)tblt.c1.390/05/17 SMI"/* SunOS4.0 1.52*/

/*
 * Mouse streams module.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/sysmacros.h>
#include <sys/signal.h>
#include <sys/user.h>
#include <sys/termios.h>
#include <sys/termio.h>
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/tty.h>
#include <sys/strtty.h>
#include <sys/systm.h>

#include <sys/kmem.h>
#include <sys/file.h>
#include <sys/uio.h>
#include <sys/errno.h>
#include <sys/reboot.h>

#include <sys/vuid_event.h>
#include "tbltio.h"
#include "tbltreg.h"

#include <sys/modctl.h>

#include <sys/cmn_err.h>

/*
 * This is the loadable module wrapper.
 */

extern struct streamtab tblt_info;

static struct fmodsw fsw = {
    "tblt",
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 2 of 36)

```
    &tblt_info,
    D_OLD
};

/*
 * Module linkage information for the kernel.
 */
extern struct mod_ops mod_strmodops;

static struct modlstrmod modlstrmod = {
    &mod_strmodops, "streams module for tablet", &fsw
};

static struct modlinkage modlinkage = {
    MODREV_1, &modlstrmod, NULL
};

static int module_keepcnt = 0;/* ==0 means the module is unloadable */

_init()
{
        return (mod_install(&modlinkage));
}

_info(modinfop)
    struct modinfo *modinfop;
{
    return (mod_info(&modlinkage, modinfop));
}

/*
 * Note: We use spl5 for the tablet because it is functionally the
 * same as spl6 and the tty mechanism is using spl5.  The original
 * code that was doing its own select processing was using spl6.
 */
#definespl_tbltspl5

#define ABS(x)((x) < 0 ? -(x) : (x))
#define BYTECLIP(x)(char)((x) > 127 ? 127 : ((x) < -128 ? -128 : (x)))

struct tbltdata {
    struct tblt_softc tbltd_softc;
    queue_t        *tbltd_readq;/* upstream read queue */
    mblk_t         *tbltd_iocpending;/* "ioctl" awaiting buffer */
    int             tbltd_flags;/* random flags */
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 3 of 36)

```
    int             tbltd_iocid;/* ID of "ioctl" being waited for */
    int             tbltd_iocerror;/* error return from "ioctl" */
    char            tbltd_oldbutt;/* button state at last sample */
    char            tbltd_oldpressure;/* pressure level at last sample */
    char            tbltd_oldproximity;/* pressure level at last sample */
    short           tbltd_state;/* state counter for input routine */
    short           tbltd_jitter;
    int     tbltd_timeout_id;/* id returned by timeout() */
    short    tbltd_pen_state;
    int     tbltd_baud_rate;
};

#define TBLT_DEV_MSC0
#define TBLT_DEV_XEROX1
#define TBLT_DEV_PEN2

#define TBLT_PEN_OUTOFPROX0
#define TBLT_PEN_ENTERPROX1
#define TBLT_PEN_INPROX2

#defineTBLT_OPEN0x00000001/* tablet is open for business */
#defineTBLT_IOCWAIT0x00000002/* "open" waiting for ioctl to finish */
#defineTBLT_IOCTOSS0x00000004/* Toss ioct return */

/*
 * Input routine states. See tbltinput().
 */
#defineTBLT_WAIT_BUTN0
#defineTBLT_WAIT_X1
#defineTBLT_WAIT_Y2
#defineTBLT_WAIT_X23
#defineTBLT_WAIT_Y24
#define TBLT_PWAIT_LEAD5
#define TBLT_PWAIT_X16
#define TBLT_PWAIT_X27
#define TBLT_PWAIT_Y8
#define TBLT_PWAIT_Y19
#define TBLT_PWAIT_Y210
#define TBLT_PWAIT_SWITCHES11

int             tblt_overrun_msg;/* Message when overrun circular buffer */
int             tblt_overrun_cnt;/* Increment when overrun circular buffer */

/*
 * Max pixel delta of jitter controlled. As this number increases the jumpiness
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 4 of 36)

```
 * of the tblt increases, i.e., the coarser the motion for medium speeds.
 */
int             tlbt_jitter_thresh = 0;

/*
 * tlbt_jitter_thresh is the maximum number of jitters suppressed. Thus,
 * hz/tlbt_jitter_thresh is the maximum interval of jitters suppressed. As
 * tlbt_jitter_thresh increases, a wider range of jitter is suppressed.
However,
 * the more inertia the tablet seems to have, i.e., the slower the tablet is to
 * react.
 */

/*
 * Measure how many (tlbt_speed_count) tblt deltas exceed threshold
 * (tlbt_speedlimit). If tlbt_speedlaw then throw away deltas over
tlbt_speedlimit.
 * This is to keep really bad mice that jump around from getting too far.
 */
int             tlbt_speedlimit = 48;
int             tlbt_speedlaw = 0;
int             tlbt_speed_count;
int             tlbtjitterrate = 12;

#defineJITTER_TIMEOUT (hz/tlbtjitterrate)

int            tbltjittertimeout;/* Timeout used when tblttimeout in effect */

/*
 * Mouse buffer size in bytes.  Place here as variable so that one could
 * massage it using adb if it turns out to be too small.
 */
int             TBLT_BUF_BYTES = 4096;

int             TBLT_DEBUG;
int             DARYL = 1;

/*
 * Pen/tablet initialization string
 */
chartblt_pen_init_press[] = "SP\rPH1\rAS1\rSU2\rAL2\rIT2\rST\r";
chartblt_pen_init_puck[] = "SP\rPH0\rAS1\rSU2\rAL2\rIT2\rST\r";

/*
 * Most of these should be "void", but the people who defined the "streams"
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 5 of 36)

```
 * data structures for S5 didn't understand datatypes.
 */
static int tbltopen(queue_t *, int, int, int);
static int tbltclose(queue_t *);
static int tbltwput(queue_t *, mblk_t *);
static int tbltrput(queue_t *, mblk_t *);
static int tbltrserv(queue_t *);

static struct module_info tbltmiinfo = {
    0,
    "tblt",
    0,
    INFPSZ,
    2048,
    128
};

static struct qinit tbltrinit = {
    tbltrput,
    tbltrserv,
    tbltopen,
    tbltclose,
    NULL,
    &tbltmiinfo
};

static struct module_info tbltmoinfo = {
    0,
    "tblt",
    0,
    INFPSZ,
    2048,
    128
};

static struct qinit tbltwinit = {
    tbltwput,
    NULL,
    tbltopen,
    tbltclose,
    NULL,
    &tbltmoinfo
};

struct streamtab tblt_info = {
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 6 of 36)

```
    &tbltrinit,
    &tbltwinit,
    NULL,
    NULL,
};

static voidtbltresched(long);
static voidtbltreioctl(long);
static voidtbltioctl(queue_t *, mblk_t *);
static int tblt_getparms(register Tblt_parms *, struct tbltdata *);
static int tblt_setparms(register Tblt_parms *, struct tbltdata *);
static voidtbltflush(struct tbltdata *);
static void tbltinput(register struct tbltdata *, char);
static voidtbltincr(struct tbltdata *);
static void tblt_abs_pos(register struct tbltdata *);
static void tblt_rel_pos(register struct tbltdata *);
static void tblt_switches(register struct tbltdata *, char);
static int getnextbaudrate(int);
static void tblt_pen_set_switch_mode(queue_t *, register struct tbltdata *,
int);
static void copy(char *, char *, int);

extern void uniqtime();

/*
 * Open a tablet.
 */
/*ARGSUSED*/
static int
tbltopen(queue_t *q,int dev, int oflag, int sflag)
{
    register struct tabletbuf *b;
    register struct tblt_softc *tblt;
    register struct tbltdata *tbltd;
    mblk_t         *mp;
    mblk_t         *set_msg;
    mblk_t         *datap;
    register struct iocblk *iocb;
    register struct termios *cb;
#ifndef _STYPES
    cred_t *crp;
#endif
    if (q->q_ptr != NULL)
    return (0);/* already attached */
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 7 of 36)

```
    if (sflag != MODOPEN)
    return (OPENFAIL);

    /*
     * Allocate an tbltdata structure.
     */
    tbltd = (struct tbltdata *)kmem_zalloc(sizeof (struct tbltdata), 0);

    /*
     * Set up queue pointers, so that the "put" procedure will accept
     * the reply to the "ioctl" message we send down.
     */
    q->q_ptr = (caddr_t)tbltd;
    WR(q)->q_ptr = (caddr_t)tbltd;

    /*
     * Setup tty modes.
     */
    if ((mp = allocb(sizeof (struct iocblk), BPRI_HI)) == NULL) {
    if (strwaitbuf(sizeof (struct iocblk), BPRI_HI))
        return(OPENFAIL);
    }
    if ((datap = allocb(sizeof (struct termios), BPRI_HI)) == NULL) {
    if (strwaitbuf(sizeof (struct termios), BPRI_HI)) {
        freemsg(mp);
        return(OPENFAIL);
    }
    }
    iocb = (struct iocblk *)mp->b_wptr;
#ifndef _STYPES
    iocb->ioc_cr = crp = crget();
#endif
    iocb->ioc_cmd = TCSETSF;
    iocb->ioc_uid = 0;
    iocb->ioc_gid = 0;
    iocb->ioc_id = getiocseqno();
    iocb->ioc_count = sizeof (struct iocblk);
    iocb->ioc_error = 0;
    iocb->ioc_rval = 0;
    mp->b_wptr += (sizeof *iocb)/(sizeof *datap->b_wptr);
    mp->b_datap->db_type = M_IOCTL;
    cb = (struct termios *)datap->b_wptr;
    cb->c_iflag = 0;
    cb->c_oflag = 0;
    cb->c_cflag = CREAD|CS8|B9600;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 8 of 36)

```
    cb->c_lflag = 0;
    bzero((caddr_t)cb->c_cc, NCCS);
    datap->b_wptr += (sizeof *cb)/(sizeof *datap->b_wptr);
    datap->b_datap->db_type = M_DATA;
    mp->b_cont = datap;
    tbltd->tbltd_flags |= TBLT_IOCWAIT;/* indicate that we're waiting for */
    tbltd->tbltd_iocid = iocb->ioc_id;/* this response */
    putnext(WR(q), mp);

    /*
     * Now wait for it.  Let our read queue put routine wake us up
     * when it arrives.
     */
    while (tbltd->tbltd_flags & TBLT_IOCWAIT) {
    if (sleep((caddr_t)&tbltd->tbltd_iocerror, STOPRI|PCATCH)) {
        ttolwp(curthread)->lwp_error = EINTR;
        goto error;
    }
    }
    if (ttolwp(curthread)->lwp_error = tbltd->tbltd_iocerror)
    goto error;

    /*
     * Set up private data.
     */
    tbltd->tbltd_state = TBLT_WAIT_BUTN;
    tbltd->tbltd_pen_state = TBLT_PEN_OUTOFPROX;
    tbltd->tbltd_readq = q;
    tbltd->tbltd_iocpending = NULL;
    tbltd->tbltd_baud_rate = B9600;
    tblt = &tbltd->tbltd_softc;

    /*
     * Allocate buffer and initialize data.
     */
    if (tblt->tblt_buf == 0) {
    tblt->tblt_bufbytes = TBLT_BUF_BYTES;
    b = (struct tabletbuf *) kmem_zalloc((u_int)tblt->tblt_bufbytes, 0);
    if (b == 0) {
        ttolwp(curthread)->lwp_error = EINVAL;
        goto error;
    }
    b->mb_size = 1 + (tblt->tblt_bufbytes - sizeof (struct tabletbuf))
        / sizeof (struct tabletinfo);
    tblt->tblt_buf = b;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 9 of 36)

```
    tblt->tblt_vuidaddr = vuid_first(TABLET_DEVID);
    tblt->pen_prev_x = 0;
    tblt->pen_prev_y = 0;
    tblt->pen_flags = 0;
    tblt->pen_nabs_scale = TBLT_PEN_NLABS_SCALE;
    tblt->pen_dabs_scale = TBLT_PEN_DLABS_SCALE;
    tblt->pen_nrel_scale = TBLT_PEN_NREL_SCALE;
    tblt->pen_drel_scale = TBLT_PEN_DREL_SCALE;
    tblt->pen_press_off_level = 10;
    tblt->pen_press_on_level = 10;
    tblt->pen_dev_setting = TBLT_PEN_DEV_TOGGLE;
    tblt->pen_dev_type= TBLT_PEN_DEV_PRESS;
    if (set_msg=allocb(sizeof(tblt_pen_init_puck),BPRI_MED)){
        copy((char *) (set_msg->b_rptr),
            tblt_pen_init_puck,
            sizeof(tblt_pen_init_puck));
        set_msg->b_wptr += sizeof(tblt_pen_init_puck);
        putnext(WR(q),set_msg);
    } else {
        return(OPENFAIL);
    }
    tbltjittertimeout = JITTER_TIMEOUT;
    tbltflush(tbltd);
    }

    tbltd->tbltd_flags = TBLT_OPEN;
#ifdef PEN_PROTOTYPE
#ifdef AUDIBLE_TIP
    tblt->pen_audible_tip = 0;
#endif
#endif

    /*
     * Tell the module below us that it should return input immediately.
     */
    (void) putnextctl1(WR(q), M_CTL, MC_SERVICEIMM);

#ifndef _STYPES
    crfree(crp);
#endif
    module_keepcnt++;/* not unloadable */
    return (0);

error:
#ifndef _STYPES
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 10 of 36)

```
    crfree(crp);
#endif
    bzero((caddr_t) tbltd, sizeof (*tbltd));
    bzero((caddr_t) tblt, sizeof (*tblt));
    return (OPENFAIL);
}

/*
 * Close the tablet
 */
static int
tbltclose(queue_t *q)
{
    register struct tbltdata *tbltd = (struct tbltdata *)q->q_ptr;
    register struct tblt_softc *tblt;

    /*
     * Tell the module below us that it need not return input immediately.
     */
    (void) putnextctl1(q, M_CTL, MC_SERVICEDEF);

    /*
     * Since we're about to destroy our private data, turn off
     * our open flag first, so we don't accept any more input
     * and try to use that data.
     */
    tbltd->tbltd_flags = 0;

    if (tbltd->tbltd_iocpending != NULL) {
    /*
     * We were holding an "ioctl" response pending the
     * availability of an "mblk" to hold data to be passed up;
     * another "ioctl" came through, which means that "ioctl"
     * must have timed out or been aborted.
     */
    freemsg(tbltd->tbltd_iocpending);
    tbltd->tbltd_iocpending = NULL;
    }
    tblt = &tbltd->tbltd_softc;
    /* Free tablet buffer */
    if (tblt->tblt_buf != NULL)
    kmem_free((void *)tblt->tblt_buf, (u_int)tblt->tblt_bufbytes);
    /* Free tbltdata structure */
    kmem_free((void *)tbltd, sizeof (*tbltd));
    module_keepcnt--;/* might be unloadable */
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 11 of 36)

```
}

/*
 * Read queue service routine.
 * Turn buffered tablet events into stream messages.
 */
static int
tbltrserv(register queue_t *q)
{
    struct tbltdata *tbltd = (struct tbltdata *)q->q_ptr;
    register struct tblt_softc *tblt;
    register struct tabletbuf *b;
    register struct tabletinfo *mi;
    register int    button_number;
    register int    hwbit, pri;
    mblk_t         *bp;

    hwbit = 0;
    tblt = &tbltd->tbltd_softc;
    b = tblt->tblt_buf;
    pri = spl_tblt();
    while (canputnext(q) && tblt->tblt_oldoff != b->mb_off) {
    mi = &b->mb_info[tblt->tblt_oldoff];
    switch (tblt->tblt_readformat) {

    case TBLT_3BYTE_FORMAT: {
        register char *cp;

        if ((bp = allocb(3, BPRI_HI)) != NULL) {
        cp = (char *)bp->b_wptr;

        *cp++ = 0x80 | mi->mi_buttons;
        /* Update read buttons */
        tblt->tblt_prevbuttons = mi->mi_buttons;

        *cp++ = mi->mi_x;
        *cp++ = -mi->mi_y;
        /* lower pri to avoid tablet droppings */
        (void) splx(pri);
        bp->b_wptr = (u_char *)cp;
        putnext(q, bp);
        pri = spl_tblt();
        } else {
        if (bufcall(3, BPRI_HI, tbltresched, (long)tbltd))
            return;/* try again later */
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 12 of 36)

```
        /* bufcall failed; just pitch this event */
        /* or maybe flush queue? */
        }
        tblt->tblt_oldoff++;/* next event */
        if (tblt->tblt_oldoff >= b->mb_size)
        tblt->tblt_oldoff = 0;/* circular buffer wraparound */
        break;
    }

    case TBLT_VUID_FORMAT: {
        register Firm_event *fep;

        bp = NULL;
        switch (tblt->tblt_eventstate) {

        case EVENT_X:
        /*
         * Send x if changed.
         */
        if ((mi->mi_x != 0) || (tblt->pen_flags & TBLT_PEN_ABSOLUTE)) {
            if ((bp = allocb(sizeof (Firm_event), BPRI_HI)) != NULL) {
            fep = (Firm_event *)bp->b_wptr;
                if (tblt->pen_flags & TBLT_PEN_ABSOLUTE) {
                fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
                  vuid_id_offset(TBLT_X_ABSOLUTE);
                fep->pair_type = FE_PAIR_DELTA;
                fep->pair = TBLT_X_DELTA;
                fep->value = mi->mi_pen_x;
            } else {
                fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
                  vuid_id_offset(TBLT_X_DELTA);
                fep->pair_type = FE_PAIR_ABSOLUTE;
                fep->pair = TBLT_X_ABSOLUTE;
                fep->value = mi->mi_x;
            }
            fep->time = mi->mi_time;
            } else {
            if (bufcall(sizeof (Firm_event), BPRI_HI, tbltresched,
              (long)tbltd))
                return;/* try again later */
            /* bufcall failed; just pitch this event */
            /* or maybe flush queue? */
            tblt->tblt_eventstate = EVENT_BUT3;
            }
        }
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 13 of 36)

```
        break;

        case EVENT_Y:
        /*
         * Send y if changed.
         */
        if ((mi->mi_y != 0) || (tblt->pen_flags & TBLT_PEN_ABSOLUTE)) {
            if ((bp = allocb(sizeof (Firm_event), BPRI_HI)) != NULL) {
            fep = (Firm_event *)bp->b_wptr;
                if (tblt->pen_flags & TBLT_PEN_ABSOLUTE) {
                fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
                  vuid_id_offset(TBLT_Y_ABSOLUTE);
                fep->pair_type = FE_PAIR_DELTA;
                fep->pair = TBLT_Y_DELTA;
                fep->value = mi->mi_pen_y;
            } else {
                fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
                  vuid_id_offset(TBLT_Y_DELTA);
                fep->pair_type = FE_PAIR_ABSOLUTE;
                fep->pair = TBLT_Y_ABSOLUTE;
                fep->value = -mi->mi_y;
            }
            fep->time = mi->mi_time;
            } else {
            if (bufcall(sizeof (Firm_event), BPRI_HI, tbltresched,
              (long)tbltd))
                return;/* try again later */
            /* bufcall failed; just pitch this event */
            /* or maybe flush queue? */
            tblt->tblt_eventstate = EVENT_BUT3;
            }
        }
        break;

        case EVENT_PRESSURE:
        /*
         * Send pressure if changed.
         */
        if (mi->mi_pressure != tblt->tblt_prevpressure) {
            if ((bp = allocb(sizeof (Firm_event), BPRI_HI)) != NULL) {
            fep = (Firm_event *)bp->b_wptr;
            fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
                  vuid_id_offset(TBLT_PRESSURE);
            fep->pair_type = FE_PAIR_NONE;
            fep->pair = 0;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 14 of 36)

```
            fep->value = mi->mi_pressure;
            tblt->tblt_prevpressure = mi->mi_pressure;
            fep->time = mi->mi_time;
            } else {
            if (bufcall(sizeof (Firm_event), BPRI_HI, tbltresched,
              (long)tbltd))
                return;/* try again later */
            /* bufcall failed; just pitch this event */
            /* or maybe flush queue? */
            tblt->tblt_eventstate = EVENT_BUT3;
            }
        }
        break;

        case EVENT_PROXIMITY:
        /*
         * Send proximity if changed.
         */
        if (mi->mi_proximity != tblt->tblt_prevproximity) {
            if ((bp = allocb(sizeof (Firm_event), BPRI_HI)) != NULL) {
            fep = (Firm_event *)bp->b_wptr;
            fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
                  vuid_id_offset(TBLT_PROXIMITY);
            fep->pair_type = FE_PAIR_NONE;
            fep->pair = 0;
            fep->value = mi->mi_proximity;
            tblt->tblt_prevproximity = mi->mi_proximity;
            fep->time = mi->mi_time;
            } else {
            if (bufcall(sizeof (Firm_event), BPRI_HI, tbltresched,
              (long)tbltd))
                return;/* try again later */
            /* bufcall failed; just pitch this event */
            /* or maybe flush queue? */
            tblt->tblt_eventstate = EVENT_BUT3;
            }
        }
        break;

        case EVENT_BUT1:
        case EVENT_BUT2:
        case EVENT_BUT3:
        /*
         * Test the button, and send an event for it if it changed.
         */
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 15 of 36)

```
        button_number = tblt->tblt_eventstate - EVENT_BUT1;
        hwbit = TBLT_HW_BUT1 >> button_number;
        if ((tblt->tblt_prevbuttons & hwbit) != (mi->mi_buttons & hwbit)) {
            if ((bp = allocb(sizeof (Firm_event), BPRI_HI)) != NULL) {
            fep = (Firm_event *)bp->b_wptr;
            fep->id = vuid_id_addr(tblt->tblt_vuidaddr) |
              vuid_id_offset(TBLT_BUTTON_1 + button_number);
            fep->pair_type = FE_PAIR_NONE;
            fep->pair = 0;
            /* Update read buttons and set value */
            if (mi->mi_buttons & hwbit) {
                fep->value = 0;
                tblt->tblt_prevbuttons |= hwbit;
            } else {
                fep->value = 1;
                tblt->tblt_prevbuttons &= ~hwbit;
            }
            fep->time = mi->mi_time;
            } else {
            if (bufcall(sizeof (Firm_event), BPRI_HI, tbltresched,
              (long)tbltd))
                return;/* try again later */
            /* bufcall failed; just pitch this event */
            /* or maybe flush queue? */
            tblt->tblt_eventstate = EVENT_BUT3;
            }
        }
        break;
        }
        if (bp != NULL) {
        /* lower pri to avoid tablet droppings */
        (void) splx(pri);
        bp->b_wptr += sizeof (Firm_event);
        putnext(q, bp);
        pri = spl_tblt();
        }
        if (tblt->tblt_eventstate == EVENT_BUT3) {
        tblt->tblt_eventstate = EVENT_X;
        tblt->tblt_oldoff++;/* next event */
        if (tblt->tblt_oldoff >= b->mb_size)
            tblt->tblt_oldoff = 0;/* circular buffer wraparound */
        } else
        tblt->tblt_eventstate++;
    }
    }
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 16 of 36)

```
    }
}

static void
tbltresched(long tbltdptr)
{
    register queue_t *q;
    register struct tbltdata *tbltd = (struct tbltdata *)tbltdptr;

    if ((q = tbltd->tbltd_readq) != 0)
    qenable(q);/* run the service procedure */
}

/*
 * Line discipline output queue put procedure: handles M_IOCTL
 * messages.
 */
static int
tbltwput(register queue_t *q, register mblk_t *mp)
{

    /*
     * Process M_FLUSH, and some M_IOCTL, messages here; pass
     * everything else down.
     */
    switch (mp->b_datap->db_type) {

    case M_FLUSH:
    if (*mp->b_rptr & FLUSHW)
        flushq(q, FLUSHDATA);
    if (*mp->b_rptr & FLUSHR)
        flushq(RD(q), FLUSHDATA);

    default:
    putnext(q, mp);/* pass it down the line */
    break;

    case M_IOCTL:
    tbltioctl(q, mp);
    break;
    }
}

static void
tbltreioctl(long tbltdptr)
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 17 of 36)

```
{
	struct tbltdata *tbltd = (struct tbltdata *)tbltdptr;
	register queue_t *q;
	register mblk_t *mp;

	if ((q = tbltd->tbltd_readq) == 0)
	return;
	if ((mp = tbltd->tbltd_iocpending) != NULL) {
	tbltd->tbltd_iocpending = NULL;/* not pending any more */
	tbltioctl(WR(q), mp);
	}
}

static void
tbltioctl(register queue_t *q, register mblk_t *mp)
{
	struct tbltdata *tbltd;
	register struct tblt_softc *tblt;
	register struct iocblk *iocp;
	caddr_t          data;
	Vuid_addr_probe *addr_probe;
	u_int            ioctlrespsize;
	int              pri;
	int              err = 0;

	tbltd = (struct tbltdata *)q->q_ptr;
	if (tbltd == 0) {
	err = EINVAL;
	goto out;
	}
	tblt = &tbltd->tbltd_softc;

	iocp = (struct iocblk *)mp->b_rptr;
	if (mp->b_cont != NULL)
	data = (caddr_t)mp->b_cont->b_rptr;

	if (TBLT_DEBUG)
	printf("tbltwput(M_IOCTL,%X,%X)\n", iocp->ioc_cmd, data);

	switch (iocp->ioc_cmd) {
	  case VUIDSFORMAT:
	if (*(int *) data == tblt->tblt_readformat)
	    break;
	if ((*(int *)data == TBLT_3BYTE_FORMAT) &&
	    (tblt->pen_flags & TBLT_PEN_ABSOLUTE)) {
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 18 of 36)

```
        cmn_err(CE_NOTE,
            "Absolute mode and 3 byte format not support\n");
        err = 1;
        break;
    }
    tblt->tblt_readformat = *(int *) data;
    /*
     * Flush tablet buffer because the messages upstream of us are in
     * the old format.
     */
    tbltflush(tbltd);
    break;

      case VUIDGFORMAT: {
    register mblk_t *datap;

    if ((datap = allocb(sizeof (int), BPRI_HI)) == NULL) {
        ioctlrespsize = sizeof (int);
        goto allocfailure;
    }
    *(int *)datap->b_wptr = tblt->tblt_readformat;
    datap->b_wptr += sizeof (int);
    mp->b_cont = datap;
    iocp->ioc_count = sizeof (int);
    break;
      }

      case VUIDSADDR:
    addr_probe = (Vuid_addr_probe *) data;
    if (addr_probe->base != VKEY_FIRST) {
        err = ENODEV;
        break;
    }
    tblt->tblt_vuidaddr = addr_probe->data.next;
    break;

      case VUIDGADDR:
    addr_probe = (Vuid_addr_probe *) data;
    if (addr_probe->base != VKEY_FIRST) {
        err = ENODEV;
        break;
    }
    addr_probe->data.current = tblt->tblt_vuidaddr;
    break;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 19 of 36)

```
      case TBLTIOGETPARMS: {
    register mblk_t *datap;

    if (TBLT_DEBUG)
        printf("tblt_getparms\n");
    if ((datap = allocb(sizeof (Tblt_parms), BPRI_HI)) == NULL) {
        ioctlrespsize = sizeof (Tblt_parms);
        goto allocfailure;
    }
    err = tblt_getparms((Tblt_parms *)datap->b_wptr, tbltd);
    datap->b_wptr += sizeof (Tblt_parms);
    mp->b_cont = datap;
    iocp->ioc_count = sizeof (Tblt_parms);
    break;
      }

      case TBLTIOSETPARMS:
    if (TBLT_DEBUG)
        printf("tblt_setparms\n");
    err = tblt_setparms((Tblt_parms *)data, tbltd);
    break;

      default:
    putnext(q, mp);/* pass it down the line */
    return;
    }

out:
    if (err != 0) {
    iocp->ioc_rval = 0;
    iocp->ioc_error = err;
    mp->b_datap->db_type = M_IOCNAK;
    } else {
    iocp->ioc_rval = 0;
    iocp->ioc_error = 0;/* brain rot */
    mp->b_datap->db_type = M_IOCACK;
    }
    qreply(q, mp);
    return;

allocfailure:
    /*
     * We needed to allocate something to handle this "ioctl", but
     * couldn't; save this "ioctl" and arrange to get called back when
     * it's more likely that we can get what we need.
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 20 of 36)

```
     * If there's already one being saved, throw it out, since it
     * must have timed out.
     */
    pri = splstr();
    if (tbltd->tbltd_iocpending != NULL)
    freemsg(tbltd->tbltd_iocpending);
    tbltd->tbltd_iocpending = mp;
    (void) splx(pri);
    (void) bufcall(ioctlrespsize, BPRI_HI, tbltreioctl, (long)tbltd);
    return;
}



static int
tblt_getparms(register Tblt_parms *data, struct tbltdata *tbltd)
{
    register struct tblt_softc *tblt = &tbltd->tbltd_softc;

    data->mode = ((tblt->pen_flags & TBLT_PEN_ABSOLUTE) ?
        TBLT_ABS_MODE : TBLT_REL_MODE);
    data->pen = tblt->pen_dev_setting;
    data->press_on = tblt->pen_press_on_level;
    data->press_off = tblt->pen_press_off_level;
#ifdef AUDIBLE_TIP
    data->audible_tip = tblt->pen_audible_tip;
#endif
    data->jitter_thresh = tlbt_jitter_thresh;
    data->speed_law = tlbt_speedlaw;
    data->speed_limit = tlbt_speedlimit;
    return 0;
}

static int
tblt_setparms(register Tblt_parms *data, struct tbltdata *tbltd)
{
    register struct tblt_softc *tblt = &tbltd->tbltd_softc;
    int err;

    err = 0;
    if (data->mode == TBLT_ABS_MODE) {
    if (tblt->tblt_readformat == TBLT_3BYTE_FORMAT) {
        cmn_err(CE_NOTE,"Absolute mode and 3 byte format not support\n");
        err = 1;
    } else {
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 21 of 36)

```
        tblt->pen_flags |= TBLT_PEN_ABSOLUTE;
    }
    } else {
    tblt->pen_flags &= ~TBLT_PEN_ABSOLUTE;
    }
    tblt->pen_dev_setting = data->pen;
    tblt->pen_press_on_level=((data->press_on >= 0) ? data->press_on : 0);
    tblt->pen_press_off_level=((data->press_off >= 0) ? data->press_off : 0);
#ifdef AUDIBLE_TIP
    tblt->pen_audible_tip = data->audible_tip;
#endif
    tlbt_jitter_thresh = data->jitter_thresh;
    tlbt_speedlaw = data->speed_law;
    tlbt_speedlimit = data->speed_limit;
    return (err);
}

static void
tbltflush(register struct tbltdata *tbltd)
{
    register struct tblt_softc *tblt = &tbltd->tbltd_softc;
    int             s = spl_tblt();
    register queue_t *q;

    tblt->tblt_oldoff = 0;
    tblt->tblt_eventstate = EVENT_X;
    tblt->tblt_buf->mb_off = 0;
    tblt->tblt_prevbuttons = TBLT_HW_BUT1 | TBLT_HW_BUT2 | TBLT_HW_BUT3;
    tblt->tblt_prevpressure = 0;
    tblt->tblt_prevproximity = 0;
    tbltd->tbltd_oldbutt = tblt->tblt_prevbuttons;
    if ((q = tbltd->tbltd_readq) != NULL && q->q_next != NULL)
    (void) putnextctl1(q, M_FLUSH, FLUSHR);
    (void) splx(s);
}

/*
 * Mouse read queue put procedure.
 */
static int
tbltrput(queue_t *q, mblk_t *mp)
{
    register struct tbltdata *tbltd = (struct tbltdata *)q->q_ptr;
    register mblk_t *bp;
    register mblk_t *imp;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 22 of 36)

```
    register mblk_t *datap;
    register struct iocblk *iocb;
    register struct termios *cb;
    register char *readp;
    struct iocblk *iocp;
#ifndef _STYPES
    cred_t *crp;
#endif

    if (tbltd == 0)
    return;

    switch (mp->b_datap->db_type) {

    case M_FLUSH:
    if (*mp->b_rptr & FLUSHW)
        flushq(WR(q), FLUSHDATA);
    if (*mp->b_rptr & FLUSHR)
        flushq(q, FLUSHDATA);

    default:
    putnext(q, mp);
    return;

    case M_BREAK:
        if (tbltd->tbltd_flags & TBLT_IOCTOSS) {
                freemsg(mp);
                return;
        }

        /*
         * Change baud rate.
         */
        if ((imp = allocb(sizeof (struct iocblk), BPRI_HI)) == NULL) {
        return;
        }
        if ((datap = allocb(sizeof (struct termios), BPRI_HI)) == NULL) {
        freemsg(imp);
        return;
        }
        tbltd->tbltd_baud_rate = getnextbaudrate(tbltd->tbltd_baud_rate);
        iocb = (struct iocblk *)imp->b_wptr;
#ifndef _STYPES
        iocb->ioc_cr = crp = crget();
#endif
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 23 of 36)

```
        iocb->ioc_cmd = TCSETSF;
        iocb->ioc_uid = 0;
        iocb->ioc_gid = 0;
        iocb->ioc_id = getiocseqno();
        iocb->ioc_count = sizeof (struct iocblk);
        iocb->ioc_error = 0;
        iocb->ioc_rval = 0;
        imp->b_wptr += (sizeof *iocb)/(sizeof *datap->b_wptr);
        imp->b_datap->db_type = M_IOCTL;
        cb = (struct termios *)datap->b_wptr;
        cb->c_iflag = 0;
        cb->c_oflag = 0;
        cb->c_cflag = CREAD|CS8|tbltd->tbltd_baud_rate;
        cb->c_lflag = 0;
        bzero((caddr_t)cb->c_cc, NCCS);
        datap->b_wptr += (sizeof *cb)/(sizeof *datap->b_wptr);
        datap->b_datap->db_type = M_DATA;
        imp->b_cont = datap;
        tbltd->tbltd_flags |= TBLT_IOCTOSS|TBLT_IOCWAIT;
        tbltd->tbltd_iocid = iocb->ioc_id;
    tbltflush(tbltd);
    flushq(q, FLUSHALL);
        putnext(WR(q), imp);
        freemsg(mp);
#ifndef _STYPES
        crfree(crp);
#endif
        return;

    case M_IOCACK:
    case M_IOCNAK:
    /*
     * If we are doing an "ioctl" ourselves, check if this
     * is the reply to that code.  If so, wake up the
     * "open" routine, and toss the reply, otherwise just
     * pass it up.
     */
    iocp = (struct iocblk *)mp->b_rptr;
    if (!(tbltd->tbltd_flags & TBLT_IOCWAIT)
      || iocp->ioc_id != tbltd->tbltd_iocid) {
        /*
         * This isn't the reply we're looking for.  Move along.
         */
        putnext(q, mp);
    } else {
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 24 of 36)

```
        tbltd->tbltd_flags &= ~TBLT_IOCWAIT;
        tbltd->tbltd_iocerror = iocp->ioc_error;
        /*
         * If we sent down a request to change the baud rate.  This
         * is the reply.  Just ignore it.
         */
        if (tbltd->tbltd_flags & TBLT_IOCTOSS) {
        tbltd->tbltd_flags &= ~TBLT_IOCTOSS;
        tbltflush(tbltd);
        flushq(q, FLUSHALL);
        } else {
            wakeup((caddr_t)&tbltd->tbltd_iocerror);
        }
        freemsg(mp);
    }
    return;

     case M_DATA:
        if (tbltd->tbltd_flags & TBLT_IOCTOSS) {
        freemsg(mp);
        return;
        }
        break;
    }

    /*
     * A data message, consisting of bytes from the tablet.
     * Hand each byte to our input routine.
     */
    bp = mp;

    do {
    readp = (char *)bp->b_rptr;
    while (readp < (char *)bp->b_wptr)
        tbltinput(tbltd, *readp++);
    bp->b_rptr = (unsigned char *)readp;
    } while ((bp = bp->b_cont) != NULL);/* next block, if any */

    freemsg(mp);
   }



/*
 * Increment the tablet sample pointer.
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 25 of 36)

```
 * Called either immediately after a sample or after a jitter timeout.
 */
static void
tbltincr(struct tbltdata  *tbltd)
{
    register struct tblt_softc *tblt = &tbltd->tbltd_softc;
    register struct tabletbuf *b;
    register struct tabletinfo *mi;
    char            oldbutt;
    register short  xc, yc;
    register int    wake;
    register int    speedlimit = tlbt_speedlimit;
    register int    xabs, yabs;
    int             s;

    b = tblt->tblt_buf;
    if (b == NULL)
    return;
    s = spl_tblt();
    mi = &b->mb_info[b->mb_off];

    if (tlbt_speedlaw) {
    xabs = ABS((int) mi->mi_x);
    yabs = ABS((int) mi->mi_y);
    if (xabs > speedlimit || yabs > speedlimit)
        tlbt_speed_count++;
    if (xabs > speedlimit)
        mi->mi_x = 0;
    if (yabs > speedlimit)
        mi->mi_y = 0;
    }

    oldbutt = mi->mi_buttons;

    xc = yc = 0;

    /* See if we need to wake up anyone waiting for input */
    wake = b->mb_off == tblt->tblt_oldoff;

    /* Adjust circular buffer pointer */
    if (++b->mb_off >= b->mb_size) {
    b->mb_off = 0;
    mi = b->mb_info;
    } else {
    mi++;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 26 of 36)

```
     }

     /*
      * If over-took read index then flush buffer so that tablet state
      * is consistent.
      */
     if (b->mb_off == tblt->tblt_oldoff) {
    if (tblt_overrun_msg)
        printf("Mouse buffer flushed when overrun.\n");
    tbltflush(tbltd);
    tblt_overrun_cnt++;
    mi = b->mb_info;
     }

     /* Remember current buttons and fractional part of x & y */
     mi->mi_buttons = oldbutt;
     mi->mi_x = (char) xc;
     mi->mi_y = (char) yc;
     if (wake)
    qenable(tbltd->tbltd_readq);/* run the service procedure */
     (void) splx(s);
}

/*
 * The input data coming from the tablet consists of a 7-byte
 * data block:
 *      Byte 1: 1, In Prox, Stylus, Pressure mode, NA, Sign-X, X15-X14
 *      Byte 2: 0, X13-X7
 *      Byte 3: 0, X6-X0
 *      Byte 4: 0, NA, NA, NA, NA, Sign-Y, Y15-Y14
 *      Byte 5: 0, Y13-Y7
 *      Byte 6: 0, Y6-Y0
 *      Byte 7: If pressure mode, then 0, Sign-pressure, P5-P0
 *                     else if Stylus, then 0, NA, NA, NA, NA, NA, Side switch,
Tip Switch
 *                        else 0, NA, NA, NA, NA, S2-S0
 *                                        S2 S1 S0        Switch
 *                                        0  0  0         None
 *                                        0  0  1         Top
 *                                        0  1  0         Left
 *                                        0  1  1         Bottom
 *                                        1  0  0         Right
 */

static void
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 27 of 36)

```
tbltinput(register struct tbltdata *tbltd, char c)
{
    register struct tblt_softc *tblt;
    register struct tabletbuf *b;
    register struct tabletinfo *mi;
    register int    jitter_radius;
    register int    temp;
    mblk_t *set_msg;


    tblt = &tbltd->tbltd_softc;
    b = tblt->tblt_buf;
    if (b == NULL)
    return;

    mi = &b->mb_info[b->mb_off];

    /*
     * This will force the first packet to be thrown away.  It is garbage
     * anyway.  Bug in hardware.
     */
    if (tbltd->tbltd_pen_state == TBLT_PEN_OUTOFPROX) {
    tbltd->tbltd_pen_state = TBLT_PEN_ENTERPROX;
    tbltd->tbltd_state = TBLT_PWAIT_LEAD;
    return;
    }

    switch (tbltd->tbltd_state) {
    case TBLT_PWAIT_LEAD:
    /*
     * Wait for a packet haeder byte.
     */
    if (!(c & TBLT_PEN_LEADBYTE)) {
        return;
    }

    /*
     * Check that we are still in proximity.  If not go back to OUTOFPROX
     */
     /*********
    if (!(c & TBLT_PEN_PROX)) {
        tbltd->tbltd_pen_state = TBLT_PEN_OUTOFPROX;
    }
    **********/
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 28 of 36)

```
    /*
     * Save control information for later use
     */
    if (c & TBLT_PEN_DEVICE) {
        tblt->pen_flags |= TBLT_PEN_DEVICE;
    } else {
        tblt->pen_flags &= ~TBLT_PEN_DEVICE;
    }
    if (c & TBLT_PEN_TIP_PRESS) {
        tblt->pen_flags |= TBLT_PEN_TIP_PRESS;
    } else {
        tblt->pen_flags &= ~TBLT_PEN_TIP_PRESS;
    }

    if ((tblt->pen_dev_setting == TBLT_PEN_DEV_PRESS_N_SIDE) ||
         (tblt->pen_dev_setting == TBLT_PEN_DEV_PRESS)) {
        if (tblt->pen_flags & TBLT_PEN_DEVICE) {
            /*
             * We have one of the two pressure pens
             */
            if (tblt->pen_dev_type != TBLT_PEN_DEV_PRESS) {
                tblt_pen_set_switch_mode(WR(tbltd->tbltd_readq), tbltd,
                    TBLT_PEN_DEV_PRESS);
                /*
                 * Just flushed queues.
                 * Wait for next lead byte
                 */
                return;
            }
        } else { /* We have the puck */
            if (tblt->pen_dev_type != TBLT_PEN_DEV_TOGGLE) {
                tblt_pen_set_switch_mode(WR(tbltd->tbltd_readq), tbltd,
                    TBLT_PEN_DEV_TOGGLE);
                /*
                 * Just flushed queues.
                 * Wait for next lead byte
                 */
                return;
            }
        }
    } else if ((tblt->pen_dev_setting == TBLT_PEN_DEV_TOGGLE_N_SIDE) ||
                (tblt->pen_dev_setting == TBLT_PEN_DEV_TOGGLE)) {
        if (tblt->pen_dev_type != TBLT_PEN_DEV_TOGGLE) {
            /*
             * We have a puck and or toggle tip pen.
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 29 of 36)

```
             */
            tblt_pen_set_switch_mode(WR(tbltd->tbltd_readq), tbltd,
                TBLT_PEN_DEV_TOGGLE);
            /*
             * Just flushed queues.
             * Wait for next lead byte
             */
            return;
        }
    }
    if (c & TBLT_PEN_PROX)
        mi->mi_proximity = 1;
        else
        mi->mi_proximity = 0;
    mi->mi_pen_x = (unsigned int)(c & TBLT_PEN_X) << X_SHIFT;
    uniqtime(&mi->mi_time);/* record time when sample arrived */
    tbltd->tbltd_state = TBLT_PWAIT_X1;
    break;

    case TBLT_PWAIT_X1:
    mi->mi_pen_x |= (unsigned int)c << X1_SHIFT;
    tbltd->tbltd_state = TBLT_PWAIT_X2;
    break;

    case TBLT_PWAIT_X2:
    mi->mi_pen_x |= (unsigned int)c;
    tbltd->tbltd_state = TBLT_PWAIT_Y;
    break;

    case TBLT_PWAIT_Y:
    mi->mi_pen_y = (unsigned int)(c & TBLT_PEN_Y) << Y_SHIFT;
    tbltd->tbltd_state = TBLT_PWAIT_Y1;
    break;

    case TBLT_PWAIT_Y1:
    mi->mi_pen_y |= (unsigned int)c << Y1_SHIFT;
    tbltd->tbltd_state = TBLT_PWAIT_Y2;
    break;

    case TBLT_PWAIT_Y2:
    mi->mi_pen_y |= (unsigned int)c;
    if (tblt->pen_flags & TBLT_PEN_ABSOLUTE) {
        tblt_abs_pos(tbltd);
    } else {
        tblt_rel_pos(tbltd);
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 30 of 36)

```
    }
    tblt->pen_prev_x = mi->mi_pen_x;
    tblt->pen_prev_y = mi->mi_pen_y;
    tbltd->tbltd_state = TBLT_PWAIT_SWITCHES;
    break;

    case TBLT_PWAIT_SWITCHES:
    tbltd->tbltd_state = TBLT_PWAIT_LEAD;

    if (!(mi->mi_proximity))
        tbltd->tbltd_pen_state = TBLT_PEN_OUTOFPROX;
    else
        tbltd->tbltd_pen_state = TBLT_PEN_INPROX;

    tblt_switches(tbltd, c);

    if (tbltd->tbltd_jitter) {
        untimeout(tbltd->tbltd_timeout_id);
        tbltd->tbltd_jitter = 0;
    }
    /*
     * If we are not in absolute mode and the button did not change,
     * see if we moved less than jitter amount.
     */

    if (!(tblt->pen_flags & TBLT_PEN_ABSOLUTE) &&
        (mi->mi_buttons == tbltd->tbltd_oldbutt) &&
        (mi->mi_proximity == tbltd->tbltd_oldproximity) &&
        (mi->mi_pressure == tbltd->tbltd_oldpressure)) {
        if (mi->mi_x == 0 && mi->mi_y == 0) {
            return;/* no, position did not change - boring event */
        }

        /*
        * Did the tablet move more than the jitter threshhold?
        */
        jitter_radius = tlbt_jitter_thresh;

        if (ABS((int) mi->mi_x) <= jitter_radius &&
            ABS((int) mi->mi_y) <= jitter_radius) {
            /*
            * Mouse moved less than the jitter threshhold. Don't
            * indicate an event; keep accumulating motions.
            * After "tbltjittertimeout" ticks expire, treat the
            * accumulated delta as the real delta.
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 31 of 36)

```
            */
            tbltd->tbltd_jitter = 1;
            tbltd->tbltd_timeout_id =
            timeout((void (*)())tbltincr, (caddr_t) tbltd,
                tbltjittertimeout);
            return;
        }
    }
    tbltd->tbltd_oldbutt = mi->mi_buttons;
    tbltd->tbltd_oldpressure = mi->mi_pressure;
    tbltd->tbltd_oldproximity = mi->mi_proximity;
    tbltincr(tbltd);
    break;
    default:
    /*
     * Bad state, lets wait for a lead byte
     */
    tbltd->tbltd_state = TBLT_PWAIT_LEAD;
    }
}

static void
tblt_abs_pos(register struct tbltdata *tbltd)
{
    register struct tblt_softc *tblt;
    register struct tabletbuf *b;
    register struct tabletinfo *mi;

    tblt = &tbltd->tbltd_softc;
    b = tblt->tblt_buf;
    if (b == NULL)
        return;

    mi = &b->mb_info[b->mb_off];

    /*
     * Scale tablet
     */
    mi->mi_pen_x = mi->mi_pen_x * tblt->pen_nabs_scale / tblt->pen_dabs_scale;
    mi->mi_pen_y = mi->mi_pen_y * tblt->pen_nabs_scale / tblt->pen_dabs_scale;

    mi->mi_pen_x -= TBLT_PEN_XSTART;
    if (mi->mi_pen_x > TBLT_PEN_XMAX) mi->mi_pen_x = TBLT_PEN_XMAX;
    else if (mi->mi_pen_x < 0) mi->mi_pen_x = 0;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 32 of 36)

```
    mi->mi_pen_y -= TBLT_PEN_YSTART;
    if (mi->mi_pen_y > TBLT_PEN_YMAX) mi->mi_pen_y = TBLT_PEN_YMAX;
    else if (mi->mi_pen_y < 0) mi->mi_pen_y = 0;
}

static void
tblt_rel_pos(register struct tbltdata *tbltd)
{
    register struct tblt_softc *tblt;
    register struct tabletbuf *b;
    register struct tabletinfo *mi;

    tblt = &tbltd->tbltd_softc;
    b = tblt->tblt_buf;
    if (b == NULL)
        return;

    mi = &b->mb_info[b->mb_off];

    /*
     * Scale tablet
     */
    mi->mi_pen_x = mi->mi_pen_x * tblt->pen_nrel_scale / tblt->pen_drel_scale;
    mi->mi_pen_y = mi->mi_pen_y * tblt->pen_nrel_scale / tblt->pen_drel_scale;

    if (tbltd->tbltd_pen_state == TBLT_PEN_ENTERPROX) {
        mi->mi_x = 0;
        mi->mi_y = 0;
        tblt->pen_prev_x = mi->mi_pen_x;
        tblt->pen_prev_y = mi->mi_pen_y;
        return;
    }

    mi->mi_x = mi->mi_pen_x - tblt->pen_prev_x;
    mi->mi_y = mi->mi_pen_y - tblt->pen_prev_y;
}

static void
tblt_switches(register struct tbltdata *tbltd, char button)
{
    register struct tblt_softc *tblt;
    register struct tabletbuf *b;
    register struct tabletinfo *mi;
    mblk_t *bp;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 33 of 36)

```
    tblt = &tbltd->tbltd_softc;
    b = tblt->tblt_buf;
    if (b == NULL)
    return;

    mi = &b->mb_info[b->mb_off];

    if (tblt->pen_flags & TBLT_PEN_DEVICE) {
    if (tblt->pen_flags & TBLT_PEN_TIP_PRESS) {
            /*
             * Invert and shift so range is between 0 and 0x40
             */
           if (button & TBLT_PEN_SIGN)
             button = (button + 0x20) & 0x3f;
             else
             button += 0x20;
            /*
             * Determine if side switch is pressed for menu
             * if we have the pressure pen with side switch.
             */
        if ((tblt->pen_dev_setting == TBLT_PEN_DEV_PRESS_N_SIDE) &&
                (button > 0x2a)) {
        mi->mi_buttons = (TBLT_HW_BUT1|TBLT_HW_BUT2);
        } else {
                 if (mi->mi_proximity)
                     mi->mi_pressure = button;
                 else
                     mi->mi_pressure = 0;
/***************************
        if (tblt->pen_flags & TBLT_PEN_TIP_ACTIVE) {
            if (button > tblt->pen_press_off_level) {
            mi->mi_buttons = (TBLT_HW_BUT2|TBLT_HW_BUT3);
            } else {
                mi->mi_buttons = (TBLT_HW_BUT1|
                     TBLT_HW_BUT2|TBLT_HW_BUT3);
            tblt->pen_flags &= ~TBLT_PEN_TIP_ACTIVE;
            }
             } else {
            if (button >= tblt->pen_press_on_level) {
            mi->mi_buttons = (TBLT_HW_BUT2|TBLT_HW_BUT3);
            tblt->pen_flags |= TBLT_PEN_TIP_ACTIVE;
            } else {
            mi->mi_buttons = (TBLT_HW_BUT1|
                     TBLT_HW_BUT2|TBLT_HW_BUT3);
            }
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 34 of 36)

```
        }
*************************/
        } /* TBLT_PEN_DEV_PRESS_N_SIDE */
    } /* TBLT_PEN_TIP_PRESS */
    else {
        switch (button & 0x07) {
        case 1:
        mi->mi_buttons = (TBLT_HW_BUT2 | TBLT_HW_BUT3);
        break;

        case 2:
        mi->mi_buttons = (TBLT_HW_BUT1 | TBLT_HW_BUT2);
        break;

        default:
        mi->mi_buttons = (TBLT_HW_BUT1|TBLT_HW_BUT2|TBLT_HW_BUT3);
        break;
        }
    }
    }  /* TBLT_PEN_DEVICE */
    else {
    switch (button & 0x07) {
    case 1:
    case 3:
        mi->mi_buttons = (TBLT_HW_BUT1 | TBLT_HW_BUT3);
        break;

    case 2:
        mi->mi_buttons = (TBLT_HW_BUT2 | TBLT_HW_BUT3);
        break;

    case 4:
        mi->mi_buttons = (TBLT_HW_BUT1 | TBLT_HW_BUT2);
        break;

    default:
        mi->mi_buttons = (TBLT_HW_BUT1|TBLT_HW_BUT2|TBLT_HW_BUT3);
    }
    }
}


static int
getnextbaudrate(int b)
{
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 35 of 36)

```
        switch (b) {
        case B1200:
                return(B2400);

        case B9600:
                return(B19200);

        case B19200:
    default:
                return(B1200);

        }
}

static void
tblt_pen_set_switch_mode(queue_t *q, register struct tbltdata *tbltd, int
mode)
{
    mblk_t *set_msg;
    register struct tblt_softc *tblt;

    tblt = &tbltd->tbltd_softc;

    switch (mode) {
    case TBLT_PEN_DEV_TOGGLE:
        if (set_msg=allocb(sizeof(tblt_pen_init_puck),BPRI_MED)){
            copy((char *) (set_msg->b_rptr),
                tblt_pen_init_puck,
                sizeof(tblt_pen_init_puck));
            set_msg->b_wptr += sizeof(tblt_pen_init_puck);
            tblt->pen_dev_type= TBLT_PEN_DEV_TOGGLE;
        } else {
            return;
        }
        break;

    case TBLT_PEN_DEV_PRESS:
        if (set_msg=allocb(sizeof(tblt_pen_init_press),BPRI_MED)){
            copy((char *) (set_msg->b_rptr),
                tblt_pen_init_press,
                sizeof(tblt_pen_init_press));
            set_msg->b_wptr += sizeof(tblt_pen_init_press);
            tblt->pen_dev_type = TBLT_PEN_DEV_PRESS;
        } else {
            return;
```

Code Example 16-13 Sample Source For Streams Module For Table Device (Sheet 36 of 36)

```
        }
        break;

    case TBLT_PEN_DEV_PRESS_N_SIDE:
        return;

    }
    putnext(q,set_msg);
    tbltflush(tbltd);
    flushq(q, FLUSHALL);
    flushq(RD(q), FLUSHALL);
}

static void
copy(char *dest, char *src, int size)
{
        while (size--)
                *dest++ = *src++;
}
```