XTLite for Macintosh
User's Guide and Getting Started Manual

XTLite is a publically available toolkit that allows you to write software that adds custom features and enhancements to QuarkXPress®. Using XTLite with QuarkXPress you will be able to:

- *Add a Word Processing Filter*
- *Add Menu Items*
- *Trap and Post User Events*
- *Create Dialog Boxes*
- *Manipulate Text*

This manual is designed to be an introduction to XTLite, as well as a tutorial for the three main features of XTLite. Step-by-step instructions are included for writing a word processing filter, adding a menu item, and trapping and posting user events in QuarkXPress with XTLite. Additional information is included at the end of this manual about QuarkXTensions® technology and the XTension® developer program.

License Agreement

This documentation and the accompanying software is made available “AS IS” without charge and without warranty of any kind.

QUARK, INC. DISCLAIMS ANY IMPLIED WARRANTY OF THIS PRODUCT INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL QUARK, INC. BE LIABLE TO A CUSTOMER FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF OR INABILITY TO USE THE SOFTWARE OR ACCOMPANYING DOCUMENTATION HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. THESE LIMITATIONS WILL APPLY EVEN IF QUARK HAS BEEN ADVISED OF SUCH POSSIBLE DAMAGES. Some states or regions do not allow the exclusion or limitation of incidental or consequential damages, or limitations on implied warranties, so the above limitations or exclusions may not apply to particular customers.

Trademark Information
Quark Publishing System, QPS and XTensions are trademarks of Quark, Inc. Quark, QuarkXPress, and QuarkCopyDesk are trademarks of Quark, Inc. which have been registered in the U.S. Patent and Trademark Office and in many other countries.

Apple Disclaimer
The following disclaimer is required by Apple Computer, Inc. It applies to Apple software. All other software is covered by the limited liability of Quark,Inc.
Apple Computer, Inc.’s (“Apple”) licensor(s) makes no warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding the software. Apple’s licensor(s) does not warrant, guarantee or make any representations regarding the software in terms of its correctness, accuracy, reliability, currentness or otherwise. The entire risks as to the results and performance of the software is assumed by you. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to you.

In no event will Apple’s licensor(s), and their directors, officers, employees or agents (collectively Apple’s licensor) be liable to you for any consequential, incidental or indirect damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising out of the use or inability to use the software even if Apple’s licensor has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you. Apple’s liability to you for actual damage from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to $50.

Contents

Contents

What is XTLite ?

For the QuarkXPress User

- **Custom Features:** With XTLite you can add custom features to QuarkXPress through software code modules called *Bulbs*.

- **Bulbs:** A Bulb is created using XTLite routines and data structures. These routines act as a hooks into QuarkXPress and facilitate communication between QuarkXPress and the Bulb you write.

- **Free:** XTLite is publically available software, provided free of charge from Quark Inc.

For the Programmer

- **Easy:** We have designed the XTLite interface to give you quick and easy programming access to some of the more common features of QuarkXPress, such as creating word processing filters, adding menu items, and handling AppleEvents.

- **Written in C:** A common programming language in use today.

- **Macintosh Code:** Any Bulb that you create with the Macintosh version of XTLite will run with QuarkXPress in emulated mode on the Power Macintosh. If you are interested in developing Bulbs that run in native mode, with the native version of QuarkXPress, see the XTLite for Power Macintosh toolkit.

- **Free:** XTLite technology is available to give you a free opportunity to experiment with QuarkXPress' underlying data structures and routines. By writing software that adds custom features and enhancements to QuarkXPress, you can see if you're interested in writing more complex software modules.

- **XTension Technology:** If you become interested in writing more complex modules, or upgrading XTLite, you may become a Quark Certified XTension Developer and receive information about and programming access to QuarkXPress through QuarkXTensions technology. The XTension interface is very similar to the Bulb interface, but incorporates more than 750 additional routines and data structures.

- **XTension Advantage:** Currently, the worldwide Quark developer network has produced over 200 QuarkXPress and Quark Publishing System® commercial XTensions.The benefits of becoming a Certified XTension Developer range from receiving free copies of Beta and Gamma Quark products to having access to an XTension marketing and supply company called XChange (for more information see the chapter titled, "Becoming an XTension Developer").

What is XTLite ?

How XTLite Works

- **In general**, XTLite Bulbs are individual code modules consisting of eleven XTLite required routines, four XTLite data structures, and any other routines you choose to add.

- **QuarkXPress** sends information to your Bulb via automatic calls to the XTLite routines. These routines are called by QuarkXPress at different times while it is running. When a routine is called, any code that you added to the body of the routine in your Bulb will be executed. When the code segment has been run, control is automatically passed back to QuarkXPress. Information can also be sent from your Bulb to QuarkXPress via the parameters in the XTLite routines. This process is invisible to the user, so your Bulb acts like an integrated part of QuarkXPress.

- **Specifically:** QuarkXPress, on launch, loads any Bulb in its home folder and calls the three XTLite routines: **setupfilter()**, **setupidle()**, and **setupmenu()**. At this time, any code in these three routines is executed. For example, in the sample Bulb MenuShell, the **setupmenu()** routine returns a value of TRUE, and passes a string "Sample XTLite Menu Item" to QuarkXPress. QuarkXPress then sets up "Sample XTLite Menu Item" as a menu item in the Utilities menu. If at some point the user selects "Sample XTLite Menu Item", QuarkXPress will call another XTLite routine **menucall()**. Any code in body of the **menucall()** routine, that you may have added to the Bulb, will then be executed.

- **Communication** between XTLite and QuarkXPress is facilitated by:

  Eleven Required XTLite Routines that can be used to:
  - Read/write text to and from QuarkXPress using a word processing filter
  - Set up and display menus
  - Link into the main event loop of QuarkXPress, to trap/post events, such as a 'mouseUp event'.

  Twelve Optional Routines that can:
  - Check the status of a text box
  - Get information about text
  - Insert and delete text
  - Set and get text attributes
  - Set and get paragraph attributes

  Four QuarkXPress Data Structures that can be used to:
  - Set text attributes such as font, face, style, size, etc.
  - Set tab justification to be LEFT, RIGHT, CENTER, and ALIGNED
  - Set leading to be relative or absolute
  - Set paragraph attributes such as indentation, justification, etc.

- **For More Information:** The remainder of this manual includes details about how the XTLite routines work, in what context they should be used, and how to obtain more powerful ways to communicate with QuarkXPress.

How to Write an XTLite Bulb

Tools you Need

- **The Macintosh XTLite toolkit:**

  - **XTLite for Macintosh User's Guide and Getting Started Manual:** the file you are currently reading.

  - **XTLite.lib:** the Think C pre-compiled library file.

  - **XTLite.h:** the Think C header file that contains a list of XTLite routines and data structures to use in your Bulb.

  - **Five Sample XTLite Bulbs:** each of which is a complete Think C project:

    Sample XTLite: adds an import/export filter

    FilterShell: adds an import/export filter

    MenuShell: demonstrates adding a menu item

    Event Shell: demonstrates trapping and posting user events

    Display Chunks: demonstrates text chunk manipulation (see the section titled "Writing a Word Processing Filter" for more information about text chunks).

  The sample XTLite Bulbs can be used to get you started. We suggest you study these examples and modify the code to make your own Bulb. For more information see the section titled "Adding Code to your Bulb".

- **QuarkXPress:** The Bulb you create will add custom features to QuarkXPress. To run your Bulb, you must have a copy of QuarkXPress for Macintosh version 3.1 or higher.

- **Think C Compiler:** The XTLite library files were compiled with Think C. We currently recommend you use this compiler to create Macintosh Bulbs.

- **Inside Macintosh Reference Library (Recommended):** Throughout the remainder of this guide references will be made to information contained in this library.

How to Write an XTLite Bulb

Things you Need to Know

- **Knowledge of the C Programming Language:** All sample Bulbs are written in C. To create your own Bulbs you will add C code to the sample Bulbs.
- **Macintosh Programming Experience:** In order to trap/post user events, and manipulate dialog boxes it is helpful to be familiar with Macintosh Toolbox calls. Almost any Macintosh Toolbox routine can be used in your Bulb. For more information see "Inside Macintosh".
- **Familiarity with the QuarkXPress Interface:** The more familiar you are with the QuarkXPress interface, the easier it will be for you to decide which features of XTLite you want to incorporate into your Bulb.

Type Definitions

- QuarkXPress uses its own type definitions that may be different from the standard C type definitions that you are used to working with. Listed below are standard C type definitions with the corresponding type definitions you should use when writing XTLite Bulbs.
- XTLite uses Macintosh Fixed notation. For more information see the section titled "Adding Code to your Bulb", or "Inside Macintosh".

Standard C Type Definitions	XTLite Type Definitions
char	int8
short,int	int16
long	int32
unsigned char	uchar
unsigned short, unsigned int	uint16
Boolean	bool8
unsigned long	uint32
short double	float64
double	float80

Data Structures - Alphabetical with Description

- **There are four** XTLite Data Structures that control Leading, Paragraph Attributes, Tab Placement, and Text Attributes. For more information about specific use of these structures see the "Writing a Word Processing Filter" section.

- **Leading Control Structure:** The structure that controls leading consists of a relative vs. absolute bit. To set the leading to relative, set this bit to 1; otherwise set it to zero, and leading will be absolute.

```
/* Leading Control Structure */

typedef struct {
        unsigned int16 relative : 1;  /* Leading is relative (vs. absolute) */
        } filterparabits;
```

- **Paragraph Attributes Structure** incorporates information from the Tab Align and Leading structures, with some additional fields.

  - The *just* field indicates the justification that should be used for this paragraph.
  - The *leftindent* and *rightindent* fields are indentation amounts from each side, in points.
  - The *firstindent* field is the amount the first line of this paragraph should be indented.
  - The *leading* field is the leading amount for this paragraph, in points.
  - The *spcbefore* and *spcafter* fields are, in points, the amount of space that should be left between this paragraph and those that precede and follow it.

```
/* Paragraph Attributes Structure */

typedef stuct {
        filterparabits a;  /* bool8 attributes from the Leading structure */
        Byte just;         /* LEFT, CENTER, RIGHT, JUSTified */
        Fixed leftindent;  /* Left Indent (relative to column/box left edge) */
        Fixed firstindent; /*First Line Indent (relative to  leftindent ) */
        Fixed rightindent; /* Right Indent(relative to column/box rightedge) */
        Fixed leading;     /* 0 means  auto  leading */
        Fixed spcbefore;   /* Space Before paragraph */
        Fixed spcafter;    /* Space After paragraph */
        filtertabspec tabs[MAXTABS]; /* User tabs from the Tab Placement struct */
} filterparaattrib;
```

- **Tab Placement Structure** controls tab justification, alignment, lead characters, and tab indents.

  - Tab justification can be set to either left, right, center or character-aligned justification using the *tabjust* field.
  - When the *tabjust* field is set to TABALIGNON, the *alignon* field can be used to set the alignment character (such as a decimal point).
  - Use the *tablead* field to insert a leader before each tab, such as spaces or dots.
  - The *tabindent* field is the amount a particular tab is indented, in points.

```
/* Tab Placement Structure */

typedef struct{
        Byte tabjust;    /* TABLEFT, TABCENTER, TABRIGHT, TABALIGNON */
        Byte alignon;                  /* byte to align on when TABALIGNON */
        unsigned int16 tablead;        /* Two characters to fill tab with */
        Fixed tabindent;               /* Offset to tab */
} filtertabspec;
```

- **Text Attributes Structure** includes information about fonts, style (face), font size, horizontal scaling, shading, kerning and tracking.

  - The *font* field contains the QuarkXPress font ID number of the text.
  - The *face* field controls style attributes such as bold, italic, etc.
  - The *size* field is a fixed number that indicates the point size of the type.
  - The *hscale* field is the horizontal scaling of the text, from 25 to 400%.
  - The *shade* field indicates the shading of the text color, from 0 to 100%.
  - The *kern* and *track* field each represent the kerning and tracking of the text, in fractional 200ths of an em space.
  - To change the value of the *font*, *size*, *hscale*, *shade*, *kern* or *track* fields, change the value field in the data structure.
  - To set a particular face attribute, logically "OR" it with the current face. For example, to set the text face to BOLD,
    textface |= BOLD;
    to turn bold off,
    textface &= ~BOLD;

```
/* Text Attributes Structure */

typedef struct {
        int16 font;        /* Font id */
        int16 face;        /* Face (style) flags */
        Fixed size;        /* Font size */
        Fixed hscale;      /* Horizt. scale factor(1.0 is 100%) */
        Fixed shade;       /* 1.0 is 100% */
        Fixed kern;        /* Fractional 200ths em */
        Fixed track;       /* Fractional 200ths em */
} filtertxtattrib;
```

Tutorials

Setting up Think C

- **Sample Projects:** If you use one of the sample Bulb projects included in this toolkit you will not need to make any of the following changes to the set up of the Think C project. The following information is provided only to explain how these sample projects were set up.

- **Set up** the project type as a Desk Accessory.

- **'vers' Resources:** XTensions should include 'vers' resources, to indicate both their own version and the version of QuarkXPress they are compatible with. The 'vers 2' resource appears directly under the Bulbs name, make sure this name says "QuarkXPress 3.1". You may change the 'vers 1' resource to provide your own Copyright information.

- **Only** 'vers 1' should be changed by you. The 'vers 2' refers to the minimum version of QuarkXPress that your Bulb will run with.

- **In order for QXP to link to your Bulb:** Your Bulb must be named so that QuarkXPress will install the proper Bulb resources for you. To do this you must:

  - Set the file creator to **XPRS** (from the "Set Project Type ..." menu in Think C).
  - Set the file type to **CUST** (from the "Set Project Type ..." menu in Think C).
  - Name your Bulb 'QuarkXTension!', so than when XPress is launched, it will add the proper resources to your Bulb
  - After XPress has loaded your Bulb once, you may change the name of the file to any thing you like.

- **Icons:** You may use a resource editor to change your Bulbs creator to **XPR3**, which will cause your XTension to be loaded by QuarkXPress version 3.1 and later.

- **On Launch** QuarkXPress looks for all files in its folder that have the resources discussed above, and identifies them as Bulbs (or XTensions). The total number of Bulbs and XTensions that QuarkXPress can handle is 50.

- **For More Information** see the reference material included with your copy of Think C, and the sample Bulbs included in this toolkit.

Tutorials

Setting up your Bulb

1. **Review the Sample Bulbs**, they illustrate how the routines should be used. Take code from the samples and use it in your own Bulb.

2. **Review the XTLite.h** header file included in this toolkit to become familiar with the QuarkXPress type definitions, XTLite routines, and the XTLite data structures.

3. **Set up your Think C Project as show above.**

4. **Include all Eleven of the Required XTLite Routines**, because QuarkXPress will expect to be able to call them. Therefore, you must include all eleven, even if you do not add code to some of them (see the sample files for an example of how this is done). If you fail to include all eleven required XTLite routines, your Bulb MAY NOT WORK!

5. **Use the Optional Routines** to help simplify the process of writing a Bulb.

6. **A Bulb is not Limited** to the eleven required and twelve optional routines, add your own routines as you see fit.

7. **Fixed Notation:** Some items in the header file use Fixed notation. These are 4-byte values that contain both the integer and fixed portions of a decimal number. Fixed values are stored as 2-byte integer part and 2-byte fractional part. When using fixed fields, the integer portion must be in the upper 16 bits of the variable. For example, to set leading to 12 points, write:

   myparaattribs.leading = 12L<<16;

   By shifting 12 to the left 16 bits, you have stored 12 in the Hiword (or integer portion) of myparaattribs.leading. See “Inside Macintosh” for additional documentation on fixed notation.

8. **Each XTLite Filter** can accommodate only one file type, so you must write a different Bulb for each file type.

9. **Each XTLite Bulb** can accommodate only one menu item, so you must write a different Bulb for each additional menu item.

10. **In order for QuarkXPress** to run your Bulb, put it in the folder with XPress. On launch QuarkXPress will automatically load your Bulb.

11. **QuarkXPress** can run a total of 50 Bulbs and XTensions. If you exceed this number, they will not be loaded correctly, and may not run.

12. **For More Information:** See the section in this document titled “Technical Support”.

Tutorials

Writing a Word Processing Filter

- **A Description and Example** of each routine appears in the section titled "Required Routines", it may be helpful to review this information before reading the following sections.

- **For More Information** see the sample Bulbs.

- **Word Processing Filters** are code modules that allow QuarkXPress to read and write a variety of text formats.

- **Across Computer Platforms:** You may write filters that read and write text across platforms (for example, read text from a Windows word processing file into the Macintosh version of QuarkXPress, provided you are using version 3.2 or higher.).

- **Add Code to the Filtershell Sample Bulb** To write a word processing filter, you will need to add your own code to the seven routines **setupfilter()**, **startread()**, **readtext()**, **endread()**, **startwrite()**, **writetext()**, and **endwrite()**; and manipulate four data structures in the Filtershell Sample Bulb.

1. **Set up your Think C Project** as shown in the previous section.

2. **Write a setupfilter() routine to install the filter.**
   - QuarkXPress will call this routine automatically once at startup.
   - Set the variables *fcreator,* and *ftype* to be the four-character Macintosh file types (for example, **TEXT**, or **XDOC**) that refer to the file creator that you want to read, and the file type of the document (see 'Inside Macintosh' for an explanation of Macintosh file types).
   - Set the variable *importok* to TRUE if you can read the file format, and set *exportok* to TRUE if you can write the file format; set them to FALSE otherwise.
   - Set the two strings, *getstr* and *savestr,* to be the strings that you want to appear in the "Get Text ..." and "Save Text..." Dialog boxes.
   - The *fextension* field is the three character file extension of a text import file (the .ABC extension from DOS). If your filter does not use DOS files, pass this value as NULL.

Example:

```
void setupfilter(OSType *fcreator,OSType *ftype,bool8 *importok,bool8 *exportok, Str255 getstr,
Str255 savestr, Str255 fextension)
{
        *fcreator = OURTEXTCREATOR;              /* set up file creator information */
        *ftype = OURTEXTTYPE;            /* set up file type information */
        *importok = *exportok = TRUE;   /* we can import and export this format */
        GetIndString(getstr,STRINGRESID,IMPORTSTRID);     /* get  Get Text  String */
        GetIndString(savestr,STRINGRESID,EXPORTSTRID); /* get  Save Text  String */

        maxtabs = maxnumoftabs();     /* always get the maximum number of tabs */
}
```

3. **Write a startread() routine**
   - When called, this routine will initialize variables and open the necessary files to start importing text. The example below shows the global storage set up and allocation that may be needed.
   - QuarkXPress will call this routine each time the user chooses “Get Text...” from the file menu, and selects your file type from the dialog box..
   - Decide what kind of global storage your filter will need, and whether you will need to allocate buffers, etc.
   - Do all setup, allocation, and initialization during **startread()**
   - On entry, *fnum* will contain the file reference number that is automatically assigned to each file. The file reference number is used a reference, don’t open this file, QuarkXPress will do this for you.

Example:

```
void startread(int16 fnum)
{
        int16 i;

        mytextattr = (filtertxtattribptr)NewPtr(sizeof(filtertxtattrib));
        myparaattr =(filterparaattribptr)NewPtr(sizeof(filterparaattrib)
+maxtabs*sizeof(filtertabspec));

        mytextattr->font = helvetica;           /* set text to 12 point helvetica */
        mytextattr->Face = PLAIN                        /* set text to plain style */
        mytextattr->size = 12L<<16; /* point size is fixed notation, so  << 16 */
        mytextattr->kern = 0L;                          /* auto kerning */
        mytextattr->shade = 0x00010000;         /* 100% shade color */
        mytextattr->hscale = 0x00010000;        /* 100% horizontal scaled text */
        mytextattr->track = 0L;

        myparaattr->relative = FALSE;           /* not relative leading */
        myparaattr->just = LEFT;                        /* left justified text */
        myparaattr->leftindent = 0L;            /* no left indent */
        myparaattr->firstindent = 0L;       /* no first line of paragraph indent */
        myparaattr->rightindent = 0L;           /* no right indent */
        myparaattr->leading = 0L;                       /* 0 means auto leading */
        myparaattr->spcbefore = 0L;     /* no space before or after paragraphs */
        myparaattr->spcafter = 0L;

        /* tabs every .5 inches (all -1L means default)...*/
        for (i = 0; i < maxtabs; i++)
                myparaattr->tabs[i].tabindent = -1L;

        thetextcolor.red = 0;                           /* text color is black */
        thetextcolor.green = 0;
        thetextcolor.blue = 0;

        filebuffer = (unsigned int8 *)NewPtr(BUFSIZE);
        remaining = doffset = 0;
}
```

4. **Write a startwrite() routine**
   - QuarkXPress will call this routine when the user chooses “Save Text...” from the File menu, and selects your text format.
   - Use this routine set up any global data structures you will need and initialize any variables.
   - Note that *fnum* is used the same way as with **startread()**.

Example:

```
void startwrite(int16 fnum)
{
        mydlog = GetNewDialog(20000,0,(WindowPtr) -1L);
        displaywind = TRUE;
        ShowWindow(mydlog);
}
```

5. **Write a readchar() routine**
   - QuarkXPress will call the **readchar()** routine to read one character at a time from the file buffer.
   - The variable *fnum* is the file reference number that is automatically assigned to the file.
   - The example code below can be used to fill a buffer with characters, and return one character at a time using **readchar().**

Example:

```
void readchar(int16 fnum)
{
        int16 err;

        if (remaining == 0) {
                        /* if buffer is empty then read a buffer full */
                remaining = BUFSIZE;
                err = FSRead(fnum,&remaining,filebuffer);
                if (err && !remaining) {
                        if (err != eofErr) return (err);
                        return (0xFF);
                }
                doffset = 0;
        }
        remaining ;
        return(filebuffer[doffset++]);
}
```

6. **Write your readtext() routine**
   - Your filter will import text using the **readtext()** routine, and will export text using the **writetext()** routine.
   - **readtext()** and **writetext()** routines will be repeatedly called as long as there is text to read or write.
   - A text run is a series of characters with identical attributes (i.e. font, size, color). A text run automatically ends when either a text attribute changes, the maximum number of 256 characters is read, or the end of a paragraph is reached, marked '\r'. If the textbuffer consists of 256 characters, it is referred to as a chunk.
   - If a text run is longer than 256 characters, return it one chunk at a time. When you have filled your buffer with a text run, set the text attributes and return the buffer (along with the character count) in textbuffer (For more information see the Display Chunks sample Bulb).
   - QuarkXPress will call the **readtext()** routine to read a text run from the source file and import it into QuarkXPress in the data structures provided.

- Import the attributes of the text using the **textattribs** and **paraattribs** data structures.
- Import the text color in the **textcolor** record as an RGB *bvalue.* QuarkXPress will convert the RGB *bvalue* into the closest matching QuarkXPress color (there are 8 such colors: red, green,blue, yellow, black, cyan, magenta, and white). For more information about the RGB structure see "Inside Macintosh".
- The *count* parameter should contain the number of characters in the *textbuffer.* When you are finished reading text, set the *count* parameter to 0.
- The *fnum* parameter contains the file reference number for use with the Macintosh File Manager routines.

Example:

```
void readtext(int16 fnum,Str255 textbuffer,int32 *count,
filtertxtattrib *textattribs,filterparaattrib *paraattribs,RGBColor *textcolor)
{
        int8 ch;
        int16 i;

        i = 0;
        while (TRUE) {
                ch = readchar(fnum); /* See readchar() above */
                /* Handle the characters you read here */
                if (ch >=     || ch == \t  || ch == \r )
                        textbuffer[i++] = ch;
                if (ch == \r  || i == 256) break;
                else if (ch == -1) break;
        }
        /* set the text attributes before returning */
        textattribs->font = mytextattr->font;
        textattribs->face = mytextattr->face;
        textattribs->size = mytextattr->size;
        textattribs->hscale = mytextattr->hscale;
        textattribs->shade = mytextattr->shade;
        textattribs->kern = mytextattr->kern;
        textattribs->track = mytextattr->track;
        paraattribs->relative = myparaattr->relative;
        paraattribs->just = myparaattr->just;
        paraattribs->leftindent = myparaattr->leftindent;
        paraattribs->firstindent = myparaattr->firstindent;
        paraattribs->rightindent = myparaattr->rightindent;
        paraattribs->leading = myparaattr->leading;
        paraattribs->spcbefore = myparaattr->spcbefore;
        paraattribs->spcafter = myparaattr->spcafter;
        BlockMove(paraattribs->tabs,myparaattr->tabs,
        maxtabs*sizeof(filtertabspec));

        textcolor->red = thetextcolor.red;
        textcolor->green = thetextcolor.green;
        textcolor->blue = thetextcolor.blue;

        *count = i;       /* tell QXP how many characters are in the text buffer */
}
```

7. **Write your writetext() routine.**
   - Use this routine to write out your text data.
   - QuarkXPress will call the **writetext()** routine as long are there are characters to be written out.
   - The *fnum* parameter contains the file reference number for use with the Macintosh File Manager routines.
   - *textbuffer* contains the characters that need to be written out.
   - *count* contains the number of characters in textbuffer.
   - *textcolor* is an RGB value that matches the color of the text being written. NOTE: *textbuffer* is NOT a Pascal string, begin writing from *textbuffer[0]*.

Example:

```
void writetext(int16 fnum,Str255 textbuffer,int32 count,filtertxtattrib *textattribs,filterparaattrib
*paraattribs,RGBColor *textcolor)
{
        int8 ch;
        int16 i,j;
        int32 size;
        int16 fserr;
        Str255 tempbuffer;

        for (j = i = 0; i < count; i++) {
                ch = textbuffer[i];
 /* Remember to filter special characters here, See the Sample XTLite Bulb */
        }
        size = j;
        fserr = FSWrite(fnum,&size,tempbuffer);
}
```

8. **Deallocate your buffers with endread().**
   - The **endread()** function is automatically called when you are finished reading text and have set the *count* parameter in **readtext()** to 0.
   - Dispose of any storage that you had previously allocated as shown in the example below.

Example:

```
void endread(int16 fnum)
{
        if (filebuffer) DisposPtr((Ptr)filebuffer);
        DisposePtr((Ptr)mytextattr);
        DisposePtr((Ptr)myparaattr);
}
```

9. **De-initialize your variables with endwrite().**
   - Call the **endwrite()** function at the end of text export to de-initialize variables.
   - Dispose of any storage that you had previously allocated as shown in the example below.

Example:

```
void endwrite(int16 fnum)
{
        int32 size;
        int16 fserr;

        DisposeDialog(mydlog);
}
```

10. **For more information see the section in this document titled "Technical Support".**

Tutorials

Adding a Menu Item

- **Items are Added to the Utilities Menu** XTLite gives you the ability to add a menu item to the Utilities menu of QuarkXPress.

- **Add Code to the MenuShell Sample Bulb** To add a menu item you will need to add your own code to the two routines **setupmenu()** and **menucall()**.

- **One Menu Item per Bulb** Each Bulb may add only one menu item.Write an additional Bulb for each menu item you need to add.

1. **Set up your Think C project as shown in the previous section.**
2. **Add your own MENUSTRID to setupmenu()**
3. **Return TRUE in setupmenu()**
   - This routine is called once at the startup of QuarkXPress to determine if your Bulb will add a menu item in the Utilities menu.
   - The variable *menustr* should contain the text, that will appear to the user, of the menu item.
   - Set **setupmenu()** to TRUE if you want to add your own menu item and FALSE otherwise.
   - The routine **GetIndString()** is a Macintosh toolbox call that gets the menu name text string from a resource.

Example:
```
bool8 setupmenu(Str255 menustr)
{
        GetIndString(menustr,STRINGRESID,MENUSTRID);    /* get  Menu  String */
        return (TRUE);
}
```

4. **Write your own menucall() routine**
   - This routine is called each time the user selects your menu item from the Utilities menu.
   - At this point, the code you write in this routine will be activated.

Example:
```
void menucall(void)
{
/* Put you own code here! */
        SysBeep(1);
}
```

5. **For more information see the section in this document titled "Technical Support".**

Trapping and Posting User Events

- **User Events can be Trapped:** The XTLite interface allows user events to be trapped before they are sent to QuarkXPress.

- **Add Code to the EventShell Sample Bulb:** XTLite traps events through the use of the routines **setupidle()** and **idlecall()**.

1. **Set up your Think C compiler as previously shown.**
2. **Set the value of setupidle() to TRUE**
   - This routine is called once at the startup of QuarkXPress to determine if your Bulb wants to handle Idle calls.
   - Set the value of **setupidle()** to TRUE if you want to handle Idle calls, and FALSE otherwise.

Example:
```
bool8 setupidle(void)
{
        return(TRUE)    /* Idle calls will be handled */
}
```

3. **Write your own idlecall() routine**
   - QuarkXPress calls this routine each time it cycles through its main event loop, which is approximately three times a second, but in some cases there may be long delays. For instance, if an alert appears, you won't receive idle calls until the user clicks OK.
   - Whatever code you have in the **idlecall()** routine will be activated during each pass.
   - The Macintosh **EventRecord** is passed into this routine, which lists the most recent user event that has occurred.
   - The example code below demonstrates the trapping of the 'mouseDown' event.
   - See "Inside Macintosh" for a complete listing of items in the **EventRecord** structure.
   - The example **idlecall()** demonstrates the trapping of the 'mouseDown' event.

Example:
```
void idlecall(EventRecord *myevent)
{
        /* Insert your code here */
        if(myevent->what == mouseDown)
                SysBeep(0);
}
```

4. **Use PPostEvent() in idlecall()**
   - To post user events, call the Macintosh toolbox routine **PPostEvent()** from **idlecall()**.
   - **PPostEvent()** will post the event and return a pointer to the event queue.
   - The following sample code demonstrates how to post the keyboard event Command 'K'.

Example:
```
#define KELLEQUIV 0x4B    /* Post a Command  K  */
        /* need variable ...*/
        EvQE1 *eventqueue;
        /*post event and return pointer to event queue... */
        PPostEvent(keyDown, (int32)KILLEQUIV, &eventqueue);
        /* set command key modifier... */
        eventqueue->evtQModifies = cmdKey;
```

5. **For more information see the section in this document titled "Technical Support".**

Routines

Required Routines

- **Include all Eleven of these Routines** in your Bulb, even if you do not add any code to them (for more information see the section titled "Adding Code to your Bulb").
- **Each Routine** is listed in this section alphabetically, and is categorized as either a Word Processing, Menu Item, or User Event routine.
- **For More Information** and an **example** of how to use each routines, see the corresponding section in the chapter titled "Tutorials".

endread() Word Processing

Synopsis
void endread(int16 fnum)

Description
Add code to this routine to deallocate the buffers used to read text into QuarkXPress.

QuarkXPress will automatically call this routine when you are finished reading text, and have set the *count* parameter in **readtext()** to 0.

Entry
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Exit
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Example

```
void endread(int16 fnum)
{
    if (filebuffer) DisposPtr((Ptr)filebuffer);
    DisposePtr((Ptr)mytextattr);
    DisposePtr((Ptr)myparaattr);
}
```

See Also
endwrite(), readtext()

endwrite() Word Processing

Synopsis
void endwrite(int16 fnum)

Description
Add code to this routine to deallocate the buffers used to write text from QuarkXPress.

QuarkXPress will automatically call this routine when you are finished writing text, and have set the *count* parameter in **writetext()** to 0.

Entry
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Exit
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Example

```
void endwrite(int16 fnum)
{
    int32 size;
    int16 fserr;

    DisposeDialog(mydlog);
}
```

See Also
endread(), writetext()

idlecall() User Event

Synopsis
void idlecall(EventRecord *myevent)

Description
Add your own code to this routine to trap events from **EventRecord**, and post events withthe toolbox call **PPostEvent()**.

QuarkXPress will call this routine each time it cycles through its main event loop (approximately 3 times per second, although long delays can occur).

Entry
myevent The Macintosh **EventRecord** that lists the most recent user event that has occurred.

Exit
myevent The Macintosh **EventRecord**.

Example

```
void idlecall(EventRecord *myevent)
{
    /* Insert your code here */
    if(myevent->what == mouseDown)
        SysBeep(0);
}
```

Caveats
See "Inside Macintosh" for more information about the Macintosh **EventRecord**.

See Also
setupidle()

menucall() Menu Item

Synopsis
void menucall(void)

Description
The code you add to this routine will be executed each time the user selects the menu item set up with the routine **setupmenu()**.

QuarkXPress will call this routine each time the user selects the **setupmenu()** menu item.

Entry
None.

Exit
None.

Example

```
void menucall(void)
{
/* Put you own code here! */
    SysBeep(1);
}
```

See Also
setupmenu()

readtext() Word Processing

Synopsis
void readtext(int16 fnum, Str255 textbuffer, int32 *count, filterxtattribs *textattribs, filterparaattribs *paraattribs, RGBColor *paraattribs)

Description
Use this routine to import text into QuarkXPress from a source file.

QuarkXPress will call this routine each time the user selects “Get Text...” with your file import type selected.

Entry
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Exit
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.
textbuffer This buffer of text that is to be read into QuarkXPress
count The number of characters in textbuffer.
textattribs The text attributes data structure.
paraattribs The paragraph attributes data structure.
textcolor The text color data structure (the RGB structure in “Inside Macintosh”).

Example
```
void readtext(int16 fnum,Str255 textbuffer, int32 *count,
filtertxtattrib *textattribs, filterparaattrib *paraattribs,
RGBColor *textcolor)
{
    int8 ch;
    int16 i;

    i = 0;
    while (TRUE) {
        ch = readchar(fnum); /* See readchar() above */
        /* Handle the characters you read here */
        if (ch >=    || ch == \t || ch == \r)
                textbuffer[i++] = ch;
        if (ch == \r || i == 256) break;
        else if (ch == -1) break;
    }
    /* initialize all text attributes here */
    /* initialize all paragraph attributes here */

    BlockMove(paraattribs->tabs,myparaattr->tabs,
    maxtabs*sizeof(filtertabspec));
    /* initialize all text color attributes here */

    *count = i; /* tell QXP how many characters are in the text buffer */
}
```

See Also
writetext()

setupfilter() Word Processing

Synopsis
void setupfilter (OSType *fcreator, OSType *ftype, bool8 *importok, bool8 *exportok, Str255 getstr, Str255 savestr, Str255 fextension)

Description
Add code to this routine to setup the necessary elements of your word processing filter.

QuarkXPress will call this routine once at startup to install your word processing filter.

Entry
None.

Exit

fcreator	Set this to be the four-character Macintosh file creator.
ftype	Set this to be the four-character Macintosh file type.
importok	Set this to TRUE if the filter can import text, FALSE otherwise.
exportok	Set this to TRUE if the filter can export text, FALSE otherwise.
getstr	Set this to be the string you want to appear in the "Get Text..." QuarkXPress dialog box.
savestr	Set this to be the string you want to appear in the "Save Text..." QuarkXPress dialog box.
fextension	Use this field as the three character file extension of a text import file (i.e. the .ABC extension from DOS). If your filter does not use DOS files, pass this value as NULL.

Example

```
void setupfilter(OSType *fcreator,OSType *ftype,bool8 *importok,
bool8 *exportok, Str255 getstr,Str255 savestr, Str255 fextension)
{
 *fcreator = CURTEXTCREATOR;  /* set up file creator information */
 *ftype = CURTEXTTYPE;       /* set up file type information */
  /* we can import and export this format...*/
 *importok = *exportok = TRUE;
 GetIndString(getstr,STRINGRESID,IMPORTSTRID); /*  Get Text String*/
  /*  Save Text String...*/
 GetIndString(savestr,STRINGRESID,EXPORTSTRID);
 maxtabs = maxnumoftabs();     /* get the maximum number of tabs */
}
```

Caveats
See "Inside Macintosh" for more information about Macintosh file types.

setupidle() User Event

Synopsis
bool8 setupidle(void)

Description
Use this routine to inform QuarkXPress that you want to receive idle calls.

QuarkXPress will call this routine once at startup.

Entry
None.

Exit
Function Return TRUE if your Bulb will receive idle calls (meaning the **idlecall()** routine in your Bulb will be called by QuarkXPress), and FALSE otherwise.

Example

```
bool8 setupidle(void)
{
    return(TRUE)    /* Idle calls will be handled */
}
```

Caveats
See "Inside Macintosh" for more information about receiving user events via the Macintosh **EventRecord**.

See Also
idlecall()

setupmenu() Menu Item

Synopsis
bool8 setupmenu(Str255 menustr)

Description
Use this routine to set up the name of your menu item.

QuarkXPress will call this routine on startup to put *menustr* in the Utilities menu.

Entry
None.

Exit
menustr — The string corresponding to your menu item that appears in the QuarkXPress Utilities menu.

Example

```
bool8 setupmenu(Str255 menustr)
{
    /* get Menu String... */
    GetIndString(menustr,STRINGRESID,MENUSTRID);
    return (TRUE);
}
```

Caveats
See "Inside Macintosh" for more information about the **GetIndString()** routine.

See Also
menucall()

startread() — Word Processing

Synopsis
void startread(int16 fnum)

Description
Use this routine to initialize any variables and open any necessary files to start importing text.

QuarkXPress will call this routine when the user selects "Get Text..." with your file type.

Entry
fnum — The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Exit
fnum — The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Example

```
void startread(int16 fnum)
{
    int16 i;

    mytextattr = (filtertxtattribptr)NewPtr(sizeof(filtertxtattrib));
    myparaattr =(filterparaattribptr)NewPtr(sizeof(filterparaattrib)
    +maxtabs*sizeof(filtertabspec));

    /* Initialize text attributes here ...*/

    /* Initialize paragraph attributes here ...*/

        /* tabs every .5 inches (all -1L means default) */
    for (i = 0; i < maxtabs; i++)
        myparaattr->tabs[i].tabindent = -1L;

    thetextcolor.red = 0; /* text color is black */
    thetextcolor.green = 0;
    thetextcolor.blue = 0;

    filebuffer = (unsigned int8 *)NewPtr(BUFSIZE);
    remaining = doffset = 0;
}
```

See Also
startwrite(), readtext()

startwrite() — Word Processing

Synopsis
void startwrite(int16 fnum)

Description
Use this routine to initialize any variables and open any necessary files to start exporting text.

QuarkXPress will call this routine when the user selects “Save Text...” with your file type.

Entry
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Exit
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Example

```
void startwrite(int16 fnum)
{
    mydlog = GetNewDialog(20000,0,(WindowPtr) -1L);
    displaywind = TRUE;
    ShowWindow(mydlog);
}
```

See Also
startread(), writetext()

writetext() Word Processing

Synopsis
void writetext(int16 fnum, Str255 textbuffer, int32 *count, filterxtattribs *textattribs, filterparaattribs *paraattribs, RGBColor *paraattribs)

Description
Use this routine to export text from QuarkXPress into a destination file.

QuarkXPress will call this routine each time the user selects “Save Text...” with your file export type selected.

Entry
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.

Exit
fnum The file reference number that is automatically assigned to the file, do not attempt to set or change the value of this variable.
textbuffer This buffer of text that is to be read into QuarkXPress.
count The number of characters in textbuffer.
textattribs The text attributes data structure.
paraattribs The paragraph attributes data structure.
textcolor The text color data structure (see "Inside Macintosh").

Example

```
void writetext(int16 fnum,Str255 textbuffer,int32 count,
filtertxtattrib *textattribs,filterparaattrib *paraattribs,
RGBColor *textcolor)
{
    int8 ch;
    int16 i,j;
    int32 size;
    int16 fserr;
    Str255 tempbuffer;

    for (j = i = 0; i < count; i++) {
        ch = textbuffer[i];
 /* Remember to filter special characters here ... */
 /* See the Sample XTLite Bulb */
    }
    size = j;
    fserr = FSWrite(fnum,&size,tempbuffer);
}
```

See Also
readtext(), startwrite()

Routines

Optional Routines

- **Twelve Optional Routines:** In addition to the eleven required routines that make up the XTLite Toolbox, there are twelve optional routines available. You may use these routine in your Bulb, or write your own version of these routines, as you see fit.
- **Use these routines** to manipulate text and paragraph attributes, and to work with QuarkXPress data structures.
- **For More Information** see the example below, and the sample Bulbs included in this toolkit.

deletetext()

Synopsis
bool8 deletetext(int32 offset, int32 numberofcharacters)

Description
This routine will delete text from the current text box.

Entry
offset	Set this to be the text offset amount that you want.
numberofcharacters	Set this to be the number of characters to delete.

Exit
Function Return	TRUE if the text was deleted, and FALSE otherwise.

Example
See the example at the end of this section, on page 42.

See Also
gettext(), inserttext()

getparaattribute()

Synopsis
bool8 getparaattribute(int16 whichattribute, Fixed *attribute, int32 startoffset, int32 endoffset)

Description
Use this routine to get the attributes of a paragraph.

Entry
whichattribute	The paragraph attribute you want.
attribute	Storage for the value of the attribute.
startoffset	The start of the offset.
endoffset	The end of the offset.

Exit
attribute	The value of the attribute requested in whichattribute.
Function Return	TRUE if whichattribute was returned in the attribute field; and FALSE if the paragraph attribute could not be returned or there was a conflict (i.e. more than one value for that paragraph attribute in the given text range).

Example
See the example at the end of this section, on page 42.

See Also
setparaattribute()

gettext()

Synopsis
bool8 gettext(int32 offset, int32 numberofcharacters, uchar *textstr)

Description
Use this routine to get text from the current text box.

Entry
offset	The text offset amount that you want.
numberofcharacters	The number of characters to delete.
textstr	The start of the offset.

Exit
Function Return	TRUE if the *offset*, *numberofcharacters* were returned from the current box in the string *textstr*; and FALSE otherwise.

Example
See the example at the end of this section, on page 42.

See Also
gettextinfo(), deletetext()

gettextattribute()

Synopsis
bool8 gettextattribute(int16 whichattribute, Fixed *attribute, int32 startoffset, int32 endoffset)

Description
Use this routine to get the attributes of text.

Entry
whichattribute	The text attribute that you want (i.e. T_SIZE).
attribute	Storage for the value of the attribute.
startoffset	The start of the text offset.
endoffset	The end of the text offset.

Exit
attribute	The value of the attribute requested in whichattribute.
Function Return	TRUE if *whichattribute* was returned in the attribute field; and FALSE if the text attribute could not be returned or there was a conflict (i.e. more than one value for that text attribute in the given range).

Example
See the example at the end of this section, on page 42.

See Also
settextattribute()

gettextinfo()

Synopsis
bool8 gettextinfo(int32 selectionstart, int32 selectionend, int32 totallength)

Description
Use this routine to get general information about the current text from the current text box.

Entry
None.

Exit

selectionstart	The start of the selected text.
selectionend	The end of the selected text.
totallength	The total length of the text in the story.

Example
See the example at the end of this section, on page 42.

See Also
gettext(), deletetext()

inserttext()

Synopsis
bool8 inserttext(int32 numberofcharacters,uchar *textstr, int32 offset)

Description
Use this routine to insert text into the current text box with the given offset.

Entry
numberofcharacters	The number of characters to insert.
textstr	The text string to be inserted.
offset	The text offset amount.

Exit
Function Return	TRUE if *numberofcharacters* of the string *textstr* were inserted into the current box by the given *offset* amount; and FALSE otherwise.

Example
See the example at the end of this section, on page 42.

Caveats
The variable *textstr* is not a Pascal string, start reading from *textstr[0].*

See Also
deletetext(), gettext().

istextboxcurrent()

Synopsis
bool8 istextboxcurrent(void)

Description
Use this routine to determine if a text box is currently selected.

Entry
None.

Exit
Function Return TRUE if a text box is both current selected and the current tool mode is CONTENTS mode, and FALSE otherwise.

Example
See the example at the end of this section, on page 42.

readchar()

Synopsis
void readchar(int16 fnum)

Description
Use this routine to read characters, one at a time, from a file buffer.

Entry
fnum The file reference number that is automatically assigned to the file, do not attempt to modify it.

Exit
fnum The file reference number that is automatically assigned to the file, do not attempt to modify it.

Example
See the sample Bulb Display Chunks.

See Also
gettext()

setparaattribute()

Synopsis
bool8 setparaattribute(int16 whichattribute, Fixed *attribute, int32 startoffset, int32 endoffset, bool16 redrawtext)

Description
Use this routine to set the attributes of a paragraph.

Entry

whichattribute	The paragraph attribute that you want to set (i.e. P_LEFTINDENT).
attribute	Storage for the new attribute value (i.e. 72L<<16(1 inch)).
startoffset	The start of the offset.
endoffset	The end of the offset.
redrawtext	Set to TRUE if you want the paragraph redrawn and FALSE otherwise.

Exit

Function Return	TRUE if *whichattribute* was set and FALSE if the paragraph attribute could not be set.

Example
See the example at the end of this section, on page 42.

See Also
getparaattribute()

settextattribute()

Synopsis
bool8 setparaattribute(int16 whichattribute, Fixed *attribute, int32 startoffset, int32 endoffset, bool16 redrawtext)

Description
Use this routine to set a text attribute.

Entry
whichattribute	The text attribute that you want to set (i.e. T_SIZE).
attribute	Storage for the new attribute value, (i.e. 72L<<16(1 inch)).
startoffset	The start of the offset.
endoffset	The end of the offset.
redrawtext	Set to TRUE if you want the text redrawn and FALSE otherwise.

Exit
Function Return	TRUE if *whichattribute* was set and FALSE if the text attribute could not be set.

Example
See the example at the end of this section, on page 42.

See Also
gettextattribute()

settextselection()

Synopsis
bool8 settextselection(int32 startoffset, int32 endoffset)

Description
Use this routine to set the current text selection range, and redraw the text range.

Entry
startoffset	The start of the text offset.
endoffset	The end of the text offset.

Exit
Function Return	TRUE if the text selection range was set, and FALSE if the text selection range could not be set.

Example
See the example at the end of this section, on page 42.

Caveats
The **turnofftextselection()** routine should have been called prior to this routine to turn off the same text.

See Also
turnofftextselection()

turnofftextselection()

Synopsis
bool8 turnofftextselection(void)

Description
Use this routine to turn off the current text selection range.

Entry
None.

Exit
None.

Example
See the example at the end of this section, on page 42.

Caveats
If you use this routine, always make sure that you call the **settextselection()** routine after you have finished working with text.

See Also
settextselection()

Optional Routines - Example

```
void idlecall(EventRecord *myevent)
{
        register int32 c;
        int32 start,end,amount,textlen;
        Fixed kern;
        int16 character;
        uchar ch;
        bool8 plus;

#define PLUSTRACKCHAR (30<<8)           /* the + track value character  }  */
#define MINUSTRACKCHAR (33<<8)   /* the - track value character  {  */

        character = (*myevent).message&keyCodeMask;
        if (((*myevent).what == keyDown || (*myevent).what == autoKey)
                && ((*myevent).modifiers&(controlKey+shiftKey+cmdKey)) ==
        controlKey+shiftKey+cmdKey
                && (character == PLUSTRACKCHAR || character == MINUSTRACKCHAR)) {
                if (istextboxcurrent()) {
                        gettextinfo(&start,&end,&textlen);
                        if (start != end) {
                                plus = character == PLUSTRACKCHAR;
                                amount = ((*myevent).modifiers&optionKey) ? 1L<<16:
                        10L<<16;
                                turnofftextselection();
                                for (c = start; c < end; c++) {
                                        gettext(c,1L,&ch);
                                        if (ch ==     && c != textlen) {
                                                gettextattribute(T_KERN,&kern,c,c+1);
                                                if (plus) {
                                                        kern += amount;
                                                        if
                                        (!settextattribute(T_KERN,kern,c,c+1,FALSE)) {
                                                                SysBeep(1);
                                                                break;
                                                        }
                                                }
                                                else {
                                                        kern -= amount;
                                                        if
                                        (!settextattribute(T_KERN,kern,c,c+1,FALSE)) {
                                                                SysBeep(1);
                                                                break;
                                                        }
                                                }
                                        }
                                }
                                settextselection(start,end);
                        }
                        else SysBeep(1);
                        (*myevent).what = nullEvent;
                }
        }
}
```

Technical Support for XTLite

- **Peer Support for XTLite** is available on America Online, CompuServe, and the Internet.
- **Questions,** answers, suggestions, constructive comments, and bug reports may be posted on any of these forums.

America Online

- The America Online forum for XTLite can be found by using:
  Keyword **QUARK**
- The XTLite toolkit can be found in the following location:
  Quark Software Libraries
    QuarkXTensions
      Macintosh XTLite.sea
      Power Macintosh XTLite.sea
      CopyDesk XTLite.sea
    QXP-Windows XTensions

CompuServe

- The CompuServe forum for XTLite can be found in the ***DTP Forum*** or by using the the keyword: **GO DTPFORUM**.
- The XTLite toolkit can be located in:
  DTP Forum (GO DTPFORUM)
    Mac DTP Utilities (Library 5)
      MACXTL.SEA (XTLite for Mac)
      PMCXTL.SEA (XTLite for Power Mac)
      CDKXTL.SEA (XTLite for CopyDesk)
    PC DTP Utilities (Library 6)
      XTLITE.ZIP (XTLite for PC)

AppleLink

- There is no public forum for XTLite on AppleLink.
- The XTLite toolkit can be found in the following location:
  Software Sampler
   3rd Party Demos/Updates
    Software Updates
     Companies K-R
      Quark
       Mac Software Libraries
        XTensions
          Macintosh XTLite.sea
          CopyDesk XTLite.sea
          Power Macintosh XTLite.sea

Internet

- **XTLite Listserver:** A listserver exists on the Internet as a way for developers to send and receive messages. It works as a remailing service — once you subscribe you will receive ALL electronic mail messages any other XTLite subscriber posts on the server. Likewise, any electronic mail message you post on the server will be remailed to ALL other XTLite developers that have subscribed. If you want to ask a question of a specific developer, or are uncomfortable posting your question in a widely distributed forum, please don't use this listserver. Send any private messages directly to the electronic mail account of the person you want to contact.

Internet

- **To subscribe to the XTLite Listserver**

  Send the following one line message:

  subscribe XTPD
  to...

  majordomo@csn.org

  No other special commands are required.

- **To post a message to the XTLite Listserver**

  Send your message to...
  XTPD@QUARK.COM

  No other special commands are required.

- **To remove yourself from the XTLite Listserver**

  Send the one-line message

  unsubscribe XTPD
  to...

  majordomo@csn.org

- **All commands** must be sent to MAJORDOMO and NOT the list! Send ONLY mail contributions to XTPD@quark.com The commands go in the body of the mail, and not the subject.

- **Activities** These public forums are designed to help XTLite developers share community knowledge specific to XTLite development. These activities are encouraged:

  - Questions regarding programming XTLite Bulbs for QuarkXPress.
  - Exchange of XTLite Bulbs with other XTLite developers.
  - Public discussion of ways to improve the XTLite program. Please be constructive. It is easy to complain, but far more useful to work towards a solution.

- **Inappropriate Activities** that will cause us to revoke your listserver subscription:

  - Carping, complaining, and moaning without any intention of working towards a resolution.
  - Personal messages to another developer.
  - Exchange of any copyrighted materials. We assume that any material that is exchanged over this listserver is NOT proprietary or trade secret.

- **For More Information:** Users of XTLite who may require additional assistance beyond the public domain forums can consider becoming a fully certified XTension developer.

Becoming an XTension Developer

Differences between XTLite Bulbs and XTensions

- An XTLite Bulb can access three features of QuarkXPress, an XTension can access 100% of the features of QuarkXPress.
- A Bulb communicates with QuarkXPress only when it is called, an XTension can send information to and receive information from QuarkXPress using special routines called Opcodes (there are approximately 100 Opcodes in the XTension interface).
- Each Bulb can add one menu item to the Utilities menu. An XTension can add several menu items, to any QuarkXPress menu except the File menu.
- Each Bulb can handle one file type at a time, each XTension can handle multiple file types.
- XTLite consists of eleven routines to communicate with QuarkXPress, the XTension interface consists of over 750 (see the following section for samples).
- As an XTLite developer you receive Peer technical support, XTension developers receive free technical support from Quark XTension programmers through electronic mail, fax, and phone.
- As an XTension developer you will receive a free subscription to a monthly technical newsletter, which is a compilation of regular electronic developer feeds that you will receive if you have an electronic mail account.
- XTension developers are eligible to attend Quark-sponsored training camps at both international and U.S. locations.
- XTension developers are eligible to receive Gamma and Beta releases of upcoming Quark Software.

Availability

- The XTension Developer Program is currently available for
  - QuarkXPress for Macintosh (which includes Power Macintosh)
  - QuarkXPress for Windows
  - Quark Publishing System

  These programs are managed by our corporate U.S. headquarters. See the section titled “Additional Information” for the ground mail address, electronic mail address, and FAX number.
- East Asian Japanese and Korean versions of QuarkXPress, which is handled through our Tokyo office. If you are interested in our East Asian developer program, please contact our Tokyo office:

  QMH Japan B.V., Japan Branch
  5FKHO Building
  3-14-16 Higashi
  Shibuya-ku
  Tokyo 150 Japan

Becoming an XTension Developer

- **How to become Certified**

  - Decide which program(s) you wish to write XTensions for. Developers can be certified for more than one platform.

  - If you are interested in the Macintosh, Windows, or Quark Publishing System developer program, print the Application included in this document, fill it out, enclose the appropriate payment, and mail it to:

    U.S. QuarkXTension Developer Desk
    1800 Grant Street
    Denver, CO 80203

  - Upon receiving your application, we will send you a Quark License agreement, and a ten-page brochure outlining the program.

  - When we receive your signed license agreement, we will review your application and if approved you will receive an acceptance letter followed by the "Inside QuarkXPress Developer Kit" for the platform(s) you are certified for.

- **Cost**

  The current fee (U.S.) for certification is $500. For information about the cost of East Asian versions of QuarkXPress, contact our Tokyo office at the address above.

Sample Routines

- **The XTension documentation** contains all of the information you need to write your own XTensions, including sample QuarkXTensions. Listed below are some of the types of routines that are available through the XTension interface (this is not a complete list).

- **Opcodes** are the commands issued to denote events. They allow you to initialize an XTension and add it to the Quark environment, handle menu selections, and handle Quark system events.

BOX_ACTIVATE
BOX_BYTESWAP
BOX_CLICK
BOX_COLOR
BOX_COLORIZE
BOX_CREATE
BOX_CURSOR
BOX_DEACTIVATE
BOX_DISPOSE
BOX_DUPLICATE
BOX_FONTINFO
BOX_IDLE
BOX_KEY
BOX_MINSIZE
BOX_MODIFY
BOX_MOVE
BOX_PRINTEND
BOX_PRINTHEADER
BOX_PRINTSTART
BOX_RESIZE
BOX_SHADE
BOX_STYLE
BOX_UPDATE
HIDDEN_CLICK
HIDDEN_COPY
HIDDEN_DELETION
HIDDEN_DRAWTEXT
HIDDEN_LEADING
HIDDEN_PASTE
HIDDEN_WIDTH
MISC_ABORTPRINT
MISC_ADDEPSCOLOR
MISC_BACKGROUND
MISC_BOXCOPY
MISC_CHECKFILETYPE
MISC_CHECK-PAGERANGE
MISC_CHECKPLATES
MISC_CLICKOUT
MISC_CLOSE
MISC_CONVERTEPS
MISC_CREATEEPSFONTS
MISC_DBPICT
MISC_DELETEITEM
MISC_DOCSTATSINFO
MISC_DRAGNDROP
MISC_DRAWBOX
MISC_DUPTABLEEND
MISC_DUPTABLESTART
MISC_EDITSAVE
MISC_EXTSAVEPICT
MISC_FRAME
MISC_GETCOLORSPACE
MISC_IMAGEFILENAME
MISC_INITPRINT
MISC_LAUNCHEPS
MISC_LAUNCHPRINT
MISC_NETENTITY
MISC_NETINIT
MISC_NETUPDATE
MISC_NEW
MISC_OPEN
MISC_OPENGUIDE
MISC_PRECOPYITEM
MISC_PREDELPAGES
MISC_PREDUPITEM
MISC_PREPCOLLECT
MISC_REGTEXT
MISC_REPORTXT
MISC_REVERT
MISC_REVERTPREP
MISC_SAVE
MISC_SAVEAS
MISC_SAVECOMPLETE
MISC_SAVEPREP
MISC_SENDCLIPPATH
MISC_SETPLATE
MISC_SLUGBYTESWAP
MISC_TRAPINFOHELP
PS_COMMENTS
PS_ENDDOC
PS_ENDEPS
PS_ENDPAGE
PS_ENDTIFF
PS_INITIALIZATION
PS_OPIADDITIONAL
PS_PROCSETS
PS_STARTDOC
PS_STARTEPS
XT_ADDMENUITEMS
XT_BOXDEF
XT_COLORSEP
XT_COLORSTUFF
XT_DEINIT
XT_DEINITTEXTREAD
XT_DEINITTEXTWRITE
XT_DOCOMMAND
XT_DOMENUITEM
XT_FREEMEM
XT_GETSTATUS
XT_HIDDEN
XT_IDLE
XT_INIT
XT_INITTEXTREAD
XT_INITTEXTWRITE
XT_MISC
XT_NETLISTCHANGE
XT_NETRECEIVE
XT_OPENPALETTES
XT_PRINTEPS
XT_PRINTPS
XT_READSTUFF
XT_WRITESTUFF
XT_XTCALL

Sample Routines

- **Alphanumeric Routines** handle string conversion and manipulation.With them you can convert between fixed values and strings, and copy, compare, and concatenate strings.

liucompstringpstrcmp	str2val2	strcpy
pstrconcat	str2valbuf	val2str2
pstrcpy	strcmp	val2strbuf
str2val	strconcat	

- **Error-Handling Routines** allow quick error checking and reporting. These routines allow you to add error descriptions to the error list, notify the user of errors that may have occurred, and check values entered in dialog boxes for validity.

allocerrorclearerror	irangecheck	recorderror
fieldrangecheck	istrrangecheck	seterror
giveerror	rangecheck	strrangecheck

- **Dialog and Window Routines** handle dialog boxes, palettes, and windows. These routines enable you to open windows, palettes, and dialog boxes, handle events that occur within windows, palettes and dialog boxes, handle controls, and set and retrieve values from fields.

activatewnd	getnewpaletteid	updatewnd
alertfilter	getpalettewptrs	WACTIVATE
BEGINWAP	invalditem	WAUTOZOOM
closewnd	isdocumentkind	WCHGDOCSTAT
dehilitetxt	ispalette	WCLICK
displaywindow	linedraw	WCLOSE
disposebits	locktexthilite	WCURSOR
dodialog	myalert	WDEACTIVATE
dotdotdot	openwnd	WDRAG
dotdotfullpath	outlines	WIDLE
doupdates	palettewsetup	wink
drawdisableditem	redisplaywindow	WKEY
edithilite	restorewnds	WKEYSWTCHOUT
ENDWAP	selectwindow	WOPEN
fgetfield	setcheck	WOTHER
findpalette	setditemenable	WRESIZE1
frontwindow	setfield	WRESIZE2
fsetfield	setkeywnd	WUPDATE
getfield	setmessageparams	
getnewdialog	setradio	

Sample Routines

- **Menu-Handling Routines** add and handle menu items.

allochierid	allocmenuid	makestylemenu
allocmenuiconid	deltmpcolormitem	

- **Text Routines** allow you to access the textual content of an open document. They enable you to create and delete characters, retrieve and change character attributes, and retrieve and change paragraph formats.

beginfract	getxepstuff2	xedrawall
blkcmp	getxetstuff	xedrawsel
chngpostatrib	getxetstuff2	xegetallattribs
cpos2bptr	locatenextstory	xegetattrib
create_fontlist	myfonttype	xegetinfo
cursorposition	nukedocs	xegetinfo2
delchars	putchar	xegetselrect
endfract	putchars	xegetstoryattb
extractword	recalcdoc	xegetstorylock
FCharWidth	setparafmt	xeputchar
getchar	showsel	xeputchars
getchars	storydirty	xesetattrib
getfontnameorid	textscroll	xesetcalc
getlos	updatfontlisthndl	xesetsel
getlos2	updathndlorvars	xesetstoryattb
getparafmt	xeactivate	xesetstorylock
getpattribs	xecalc	xgetcaretpos
gettattribs	xecopytext	xtgetauxfontinfo
getxepstuff	xedelchars	

- **Style Sheets and H&J Routines** allow you to access style sheets and H&J specifications. These routines enable you to: add, modify, and delete style sheets; and add, modify, and delete H&J specifications.

addhandj	findhyphexcep	getstylebyindex2
addhyphexcep	findhyphexcepbyindex	getstylebyname
addstyle	gethandjbyindex	getstylebyname2
beginhyphexcep	gethandjbyindex2	hyphenateword
counthandjs	gethandjbyindex3	numhyphexceps
countstyles	gethandjbyname	sethandjbyindex
delhandjbyindex	gethandjbyname2	sethyphmethod
delhyphexcep	gethandjbyname3	setstylebyindex
delstylebyindex	gethyphmethod	setstyleontext
endhyphexcep	getstylebyindex	

Sample Routines

- **Import and Export Routines** enable QuarkXPress to read and write new file formats from text to geometry and graphics.

addpicimport	addtextimport2	gettextpict2
addpicimport2	addtiffimport	importbuf
addtextexport	exportbuf	savetext
addtextexport2	getfilterinfo	savetext2
addtextimport	gettextpict	

- **System Routines** allow you to access system parameters at both the document and system level.

getprefs	getsysdefs2	setstepoffsets
getspacealignvalues	getsysinfo	setsysdefs
getstepoffsets	setprefs	setsysdefs2
getsysdefs	setspacealignvalues	setsysinfo

- **File-Handling Routines** give you the power to find files, save documents, and manipulate path names.

addftypealias	getfile	processtats
duplicatefile	getfilesbytype	putfile
extendedsave	getfilesbytype2	save
extractflname	getfullpath	setbackuppath
findfile	HandToXHand	setuppath2
getbackuppath	newsave	writestats

- **Network Communication Routines** exchange information between XTensions so that you can find out who is available on a network; and send and receive information from other XTensions.

getnetinfo	getruncount	qphandler
getnetinfo2	qpacketcancel	rebuildnetlist
getnetlist	qpacketsubscribe	sendq

- **Hidden Text Routines** allow XTensions to insert hidden text data into the text stream. You can use the routines to anchor elements to text (anchored graphics are a form of hidden text), and identify text so that it can be read and processed by your machine.

cleanselrange	h_delchars	increment
decrement	h_getchar	inserthidden
findopcode	h_getchars	skiphidden

Sample Routines

- **Utility Routines** provide XTensions with an array of flexible routines that allow you to simplify XTension source code, debug XTension source code, add and select tools, add menu commands, hyphenate words, obtain information about a printer, and send PostScript commands directly to a printer.

addhelp	docommand2	quitxpress
addtool	END_XT	registersecurext
ADDXT	getmode	registerxt
ADDXTSECURE	getwarrantyinfo	scrollxticon
allocicon	getxtinfo	setmode
BEGIN_XT	LOCK_XT	setundo
cmdperiod	makeeps	spellmenu
copyprintsetup	postsend	UNLOCK_XT
devtype	postsendbuf	updatedocfonts
docommand	postsendnocr	zerodata

- **Box Routines** allow you to access text and picture boxes so you can create, manipulate, and delete picture and text boxes; navigate between boxes, perform global operations on the contents of boxes, and identify boxes for current and future reference.

addframe	getscaledcontrgn	newbox2
box2page	getscaledframe	nextbox
bringforward	getscaledframerect	nexttextbox
copyitems	getslug	offsetbox
createmultbox	getslug2	pasteitems
curboxsprdorigin	getsprdbox	pasteitemsatxy
deletebox	gettextflowrgn	prevbox
disposebox	gotobox	prevtextbox
doframe	hasrunpolys	relinkbox
dohandles	inhandles	sendbehind
drawbox	installbox	setbox
fgetbbox	invalbox	setbox2
findslug	invalframe	setcurbox
findslug2	isapicture	setslug
finsetpoly	isgraphicbox	setslug2
firstbox	ismanualimage	settextoutset
free_pixmap	istextbox	simplefpoly
getbox	isuserbox	special
getbox2	lastbox	updatethebox
getboxtype	movebox	xelinkbox
getboxtypes2	newbox	xeunlinkbox

- **Box-Grouping Routines** handle relationships between boxes. Using them, you can group boxes, add boxes to a group, ungroup boxes, get information about a group, and create a multiple-box selections.

addtogroup	deletegroupbox	isboxingroup
boxesingroup	getgroupelement	recalcgpbox
constraingroup	group	ungroup

Sample Routines

- **Spread and Page Routines** handle the creation and manipulation of spreads and pages. Use them to create, move, and delete spreads and pages; navigate among spreads and pages; apply master pages; and convert spread coordinates between spread and page coordinates.

applymaster2pages	insertpages	pagept2sprd
createmaster	insertpages2	redrawpage
deldefpages	interdoccopypages	replicateitems
deletepages	isdbldef	rightmaster
getdisppagenum	mousepage	setsectionstart
getfullmastername	movepages	setsprdorigin
getmastername	nummasterpages	sprd2pages
getpagedata	nummastersprds	sprdfrect2page
getpagedata2	numpages	sprdpt2page
getpageseq	numsprds	updatepagepalette
gotomaster	offpage	whichpage
gotopage	pagefrect2sprd	xtgetpageinfo

- **Guideline Routines** give you the ability to add, move and delete page guides.

addguide	getallguides	registerguide
countguides	getallguides2	setguidebyindex
delguidebyindex	getguidebyindex	

- **Color Routines** control the color attributes in a document and enable you to add, change, and delete colors; get information about specific colors; retrieve and change tapping values, convert between RGB, CMYK, and HSB color models.

addbackground	mygetgray	shadecolor
addcolor	pantone2rgb	xtgetlistcolor
cmyk2rgb	restorecolor	xtgetlistcolor2
countcolors	rgb2cmyk	
delcolorbyid	rgb2hsv	
getcolorbyindex	setcolor	
getcolorbyname	setcolorbyid	
hsv2rgb	shadecmykcolor	

Becoming an XTension Developer

Additional Information

- **To receive more information** about the XTension Developer program, you may contact the Developer Desk by ground mail, electronic mail or Fax.

- **Electronic Mail:** You can reach the XTension Developer Desk at Quark via electronic mail in several ways. Electronic mail messages can be received via AppleLink, CompuServe, and America Online. Messages received by electronic mail will receive priority of any other means of communication.

  - Electronic Mail Address

    AppleLink **QUARKXT**

    CompuServe **75140,1136**

    Internet **75140.1136@COMPUSERVE.COM**

    Japan Developer Desk (for East Asian Developers)

    AppleLink **QUARK.J.DVJ (send ATTN: XTension East Asian Developer Desk)**

  - Ground Mail Address for all programs except East Asian

    U.S. QuarkXTension Developer Desk
    1800 Grant Street
    Denver, CO 80203

  - Ground Mail Address for a East Asian Developers

    QMH Japan B.V., Japan Branch
    5FKHO Building
    3-14-16 Higashi
    Shibuya-ku
    Tokyo 150 Japan

  - Fax Number
    (303) 894-3399

Application for the QuarkXTension Developer Program

- The complete application for the developer program is included in the enclosed file: **XTension Developer Application.** You may print the application, fill it out, and send in the appropriate payment to the address above. Upon approval you will receive the complete XTension Developer Kit which includes everything you need to create your own XTensions.