TX Text-Control

DBS

Text-Control Properties and Events

All of the Properties, Events, and Methods for the Text-Control are listed in alphabetical order in the following table.

Align Property
Alignment Property
BackColor Property
BackStyle Property
BorderStyle Property
ButtonBar Property
Change Event
Click Event
Clip Property
ClipChildren Property
ClipSiblings Property
ControlChars Property
CurrentPages Property
DblClick Event
DragDrop Event
DragIcon Property
DragMode Property
DragOver Event
Enabled Property
ErrorCode Event
FindReplace Property
FontBold Property
FontDialog Property
FontItalic Property
FontName Property
FontSize Property
FontStrikethru Property
FontUnderline Property
ForeColor Property
FormatSelection Property
FrameDistance Property
FrameLineWidth Property
FrameStyle Property
GotFocus Event
Height Property
HideSelection Property
hWnd Property
HScroll Event
Index Property
InsertionMode Property
KeyDown Event
KeyPress Event
KeyUp Event
Left Property
Language Property

Load Property
LostFocus Event
MouseDown Event
MouseMove Event
MouseUp Event
MousePointer Property
Move Event
Move Method
Name Property
PageHeight Property
PageMarginB Property
PageMarginL Property
PageMarginR Property
PageMarginT Property
PageWidth Property
ParagraphDialog Property
Parent Property
PosChange Event
PrintDevice Property
PrintPage Property
ReadOnly Property
Refresh Method
RTFExport Property
RTFImport Property
Ruler Property
Save Property
Scrollbars Property
ScrollPosX Property
ScrollPosY Property
SelLength Property
SelStart Property
SelText Property
SetFocus Method
Size Event
SizeMode Property
StatusBar Property
TabIndex Property
TabKey Property
TabStop Property
Tag Property
Text Property
TextColor Property
Top Property
Width Property
Visible Property
VScroll Event
VTSpellCheck Property
VTSpellDictionary Property
ZOrder Method

Introduction

Welcome to Text-Control for Visual Basic, the text processor in a single VBX control. Using Text-Control, you can create all kinds of text-based applications with the ease of programming that is characteristic of Visual Basic and with highly sophisticated formatting and display capabilities which are normally the exlusive domain of large word processing packages.

System Requirements

Text-Control for Visual Basic requires the following minimum configuration:

- Windows 3.1 or higher.

- Microsoft Visual Basic or any development platform which supports Visual Basic 1.0 compatible custom controls, like Microsoft Visual C++ 1.5x, Borland C++ 4.x and dBase for Windows.

Distributing your Applications

Text-Control can only be used in design mode if the license file TX4VBDEV.DLL is present in the same directory as the custom control file TX4VBL.VBX. This license file is not required for the operation of compiled programs (EXE files), and may not be distributed with your applications.

The following Text-Control files are required for a program to run and should be copied to the user's Windows system directory:

txl.dll, tx4vbl.vbx, txtools.dll, tx_rtf.dll, wndtools.dll

How this Help File is Organized

Part 1, Creating a Simple Word Processor, shows you how to create a small word processor from scratch with just a few lines of code.

Part 2, Text-Control Programming, is a guide to programming Text-Control and its tools, explaining the parts which have been omitted from part one.

Part 3, Text-Control Properties and Events, is a list of all of the Properties, Events, and Methods of Text-Control and its tools.

Appendix A, Text-Control Error Codes, is a list of Text-Control's error codes.

Appendix B, Mouse and Keyboard Assignment, describes the Text-Control keyboard and mouse interface.

Alignment Property

Description: Specifies the alignment of text in a Text-Control.

Usage: [form.]TextControl.Alignment [= *alignment*]

Remarks: The Alignment Property settings are:

Setting	Description
0	Text is left aligned. (Default)
1	Text is right aligned.
2	Text is centered.
3	Text is justified.
4	This value cannot be assigned to the Property. Its purpose is to indicate that the selected text contains paragraphs which have different types of alignment.

If the FormatSelection Property has previously been set to True, changing the Alignment Property affects only the currently selected paragaph. If FormatSelection has been set to False the setting applies to the entire control, in which case a value of 4 does not occur.

Data Type: Integer.

ButtonBar Property

Description: Specifies the button bar control to be used with a Text-Control.

Usage: [form.]TextControl.ButtonBar [= *button bar control name*]

Remarks: The Button Bar, like the Status Bar and the Ruler, is one of the additional controls which are contained in the tx4vb.vbx file. The chapter "Creating a Simple Word Processor" in this manual describes how to use these controls.

Data Type: String.

See also: Ruler Property, StatusBar Property.

Clip Property

Description: Performs Text-Control clipboard actions. Not available at design time.

Usage: [form.]TextControl.Clip = *action*

Remarks: The parameter can have one of the following values:

Setting	Description
CLIP_CUT (1)	Cut out the selected text and copy it to the clipboard.
CLIP_COPY (2)	Copy the selected text to the clipboard.
CLIP_PASTE (3)	Paste text from the clipboard.
CLIP_CLEAR (4)	Clear the selection.

Data Type: Integer.

Example: This example copies the selected text from a Text-Control named "TextControl1" to the clipboard when the user selects the "Edit/Copy" menu item:

```
Sub mnuEdit_Copy_Click ()
    TextControl1.Clip = CLIP_COPY
End Sub
```

ClipChildren Property

Description: The ClipChildren Property is only used for Text-Controls which act as a container for other Text-Controls. When ClipChildren set to True, the areas occupied by the child controls are excluded from the update area. So if transparent controls are used as children of the container control, this Property must be set to False.

Usage: [form.]TextControl.ClipChildren [= *boolean*]

Remarks: The ClipChildren Property settings are:

Setting	Description
True	Exclude areas which are occupied by child controls from the update area. (Default).
False	Update areas which are occupied by child controls.

Data Type: Boolean.

See also: ClipSiblings Property.

Example: See Forms2 sample program.

ClipSiblings Property

Description: The ClipSiblings Property determines the clipping behaviour of each of the child controls which belong to a common container control. It must be set to False if the program is to allow transparent Text-Controls to overlap other Text-Controls.

Usage: [form.]TextControl.ClipSiblings [= *boolean*]

Remarks: The ClipSiblings Property settings are:

Setting	Description
True	Excludes those areas occupied by other child controls from the update area . (Default).
False	Updates areas which are occupied by other child controls.

Data Type: Boolean.

See also: ClipChildren Property.

Example: See Forms2 sample program.

ControlChars Property

Description: Specifies if control characters are visible.

Usage: [form.]TextControl.ControlChars [= *boolean*]

Remarks: The ControlChars Property settings are:

Setting	Description
True	Control characters, like space or paragraph break, are made visible.
False	Control characters are invisible.

Data Type: Boolean.

CurrentPages Property

Description: Specifies the number of pages contained in the Text-Control. Not available at design time; read-only at run time.

Usage: [form.]TextControl.CurrentPages

Remarks: The value of this Property depends on the size of the text as well as on the settings of the PageHeight, PageWidth and PageMargin*x* Properties.

Data Type: Integer.

See also: PageHeight Property, PageWidth Property, PageMarginx Properties, PrintDevice Property, PrintPage Property, .

Example: See PrintPage Property example.

ErrorCode Event

Description: Occurs when the Text-Control reports an error.

Usage: Sub TextControl_ErrorCode(ErrorNumber As Integer)

Remarks: The error codes are listed in appendix A.

FindReplace Property

Description: Displays a "Find" or "Replace" dialog box. Not available at design time.

Usage: [form.]TextControl.FindReplace = *type of dialog*

Remarks: The Property settings are:

Setting	Description
1	Display a "Find" dialog box.
2	Display a "Replace" dialog box.

Data Type: Integer.

FontBold Property

Description: Determines the font style.

Usage: [form.]TextControl.FontBold [= *style*]

Remarks: At design time, this Property works like the standard FontBold Property. At runtime, the settings are:

Setting	Description
0	The characters are not bold.
1	The characters are bold.
2	Indicates that the selected text contains bold and non-bold characters. This can only occur if the FormatSelection Property has been set to True.

Data Type: Integer.

See also: FontItalic Property, FontStrikethru Property, FontUnderline Property, Font Dialog Property , FormatSelection Property.

FontDialog Property

Description: Invokes the Text-Control's built-in font dialog box and, after the user has closed the dialog box, specifies whether he has changed something. Not available at design time; read-only at run time.

Usage: [form.]TextControl.FontDialog

Remarks: The changes made in the dialog box apply to the currently selected text. The Property settings are:

Setting	Description
True	The user has changed one or more attibutes.
False	The formatting remains unchanged.

Data Type: Boolean.

FontItalic Property

Description: Determines the font style.

Usage: [form.]TextControl.FontItalic [= *style*]

Remarks: At design time, this Property works like the standard FontItalic Property. At runtime, the settings are:

Setting	Description
0	The characters are not italic.
1	The characters are italic.
2	Indicates that the selected text contains italic and non-italic characters. This can only occur if the FormatSelection Property has been set to True.

Data Type: Integer.

See also: FontBold Property, FontStrikethru Property, FontUnderline Property, FontDialog Property, FormatSelection Property.

FontStrikethru Property

Description: Determines the font style.

Usage: [form.]TextControl.Strikethru [= *style*]

Remarks: At design time, this Property works like the standard FontStrikethru Property. At runtime, the settings are:

Setting	Description
0	The characters are not struck through.
1	The characters are not struck through.
2	Indicates that the selected text contains struck through and non-struck through characters. This can only occur if the FormatSelection Property has been set to True.

Data Type: Integer.

See also: FontBold Property, FontItalic Property, FontStrikethru Property, FontUnderline Property, FontDialog Property, FormatSelection Property.

FontUnderline Property

Description: Determines the font style.

Usage: [form.]TextControl.FontUnderline [= *style*]

Remarks: At design time, this Property works like the standard FontUnderline Property. At runtime, the settings are:

Setting	Description
0	The characters are not underlined.
1	The characters are underlined.
2	Indicates that the selected text contains underlined and non-underlined characters. This can only occur if the FormatSelection Property has been set to True.

Data Type: Integer.

See also: FontBold Property, FontItalic Property, FontStrikethru Property, FontDialog Property, FormatSelection Property.

FormatSelection Property

Description: Specifies if character and paragraph formatting Properties apply to the whole text or to a particular selection only.

Usage: [form.]TextControl.FormatSelection

Remarks: The Properties which are affected are Alignment, FontBold, FontItalic, FontName, FontSize, FontStrikethru, and FontUnderline.

Setting	Description
True	The formatting Properties only apply to selected text. This mode works only at run time, because at design time it is not possible to select text.
False	The formatting Properties apply to the whole text. This is the default mode.

Data Type: Boolean.

FrameDistance Property

Description: Specifies the distance between text and paragraph frame for the currently selected paragraph(s). Not available at design time.

Usage: [form.]TextControl.FrameDistance [= *distance*]

Remarks: The Property value is set to -1 if the user selects two or more paragraphs which have different frame distance settings.

Data Type: Integer.

FrameLineWidth Property

Description: Specifies the line widths of the currently selected paragraph's frames. Not available at design time.

Usage: [form.]TextControl.FrameLineWidth [= *line width*]

Remarks: The Property value is set to 0 if the user selects two or more paragraphs which have different line width settings.

Data Type: Integer.

FrameStyle Property

Description: Specifies the style of the currently selected paragraph's frames. Not available at design time.

Usage: [form.]TextControl.FrameStyle [= *Style Flags*]

Remarks: The Property value can be a combination of the following flags:

Setting	Description
BF_LEFTLINE	Draws a left frame line.
BF_RIGHTLINE	Draws a right frame line.
BF_TOPLINE	Draws a top frame line.
BF_BOTTOMLINE	Draws a bottom frame line.
BF_BOX	Draws a complete box.
BF_TABLINES	Draws a vertical line at each tab position.
BF_TABLE	Draws a complete box including vertical lines at every tab position.
BF_SINGLE	Draws a single line.
BF_DOUBLE	Draws a doubled line.
BF_NOLEFTLINE	Resets an existing left line.
BF_NORIGHTLINE	Resets an existing right line.
BF_NOTOPLINE	Resets an existing top line.
BF_NOBOTTOMLINE	Resets an existing bottom line.
BF_NOTABLINES	Resets existing tabulator lines.

The Property value is set to -1 if the user selects two or more paragraphs which have different frame style settings.

Data Type: Integer.

HideSelection Property

Description: Specifies whether a text selection is to be hidden when the Text-Control window is not active.

Usage: [form.]TextControl.HideSelection [= *boolean*]

Remarks: The HideSelection Property settings are:

Setting	Description
True	The selection is hidden when the Text-Control window becomes inactive.
False	The selection stays visible.

Data Type: Boolean.

HScroll Event

Description: Occurs when the horizontal scroll position has been changed.

Usage: Sub TextControl_HScroll()

See also: VScroll Event.

InsertionMode Property

Description: Specifies Insert or Overwrite mode.

Usage: [form.]TextControl.InsertionMode [= *boolean*]

Remarks: The InsertionMode settings are:

Setting	Description
True	Insert mode.
False	Overwrite mode.

Data Type: Boolean.

Language Property

Description: Determines the language in which Text-Control displays dialog boxes and error messages. Not available at design time.

Usage: [form.]TextControl.Language = Country code

Remarks: The default language is determined by the 'iCountry=' setting in win.ini.

Setting	Description
34	Spanish
49	German
else	English

Data Type: Boolean.

Load Property

Description: Loads the contents of a text control with all text and format information from a file which has previously been saved using the Save Property. Not available at design time.

Usage: [form.]TextControl.Load = *DOS File Handle*

Remarks: The file must be opened in binary mode. The Visual Basic FileAttr() function is used to convert the Visual Basic file number to a DOS file handle.

Data Type: Integer.

See also: Save Property.

Example: This example opens the file stated in the function parameter "Filename" and loads its contents into TextControl1:

```
Sub OpenFile (Filename)
   ' open the selected file
   Open Filename For Binary As #1
   If Err Then
      MsgBox "Can't open file: " + Filename
      Exit Sub
   End If
   ' Use the FileAttr function to get a DOS file handle
   ' from the VisualBasic file number and pass it on to TX.
   TextControl1.Load = FileAttr(1, 2)
   Close #1
End Sub
```

Move Event

Description: Occurs when a Text-Control has been moved with the mouse while depressing the ALT key.

Usage: Sub TextControl_Move

See also: Size Event, SizeMode Property

PageHeight Property

Description: Specifies the height of the printer page.

Usage: [form.]TextControl.PageHeight [= *height*]

Remarks: The height of the actual printed area is PageHeight minus PageMarginB minus PageMarginT. The maximum value depends on the capabilities of the selected printer and must not exceed 32767 twips. (Twips is the default scale in Visual Basic. One Twip is a twentieth of a Point. There are 1,440 twips to one inch).

If PageHeight is 0, the Height Property is used instead. This setting can be used to place several controls without scrollbars on a page. The PageMarginT Property then determines the vertical position of the control.

Data Type: Long.

See also: PageWidth Property, PageMarginx Properties, PrintDevice Property, PrintPage Property.

Example: See PrintPage Property example.

PageMarginB Property

Description: Specifies the bottom margin on the printed page.

Usage: [form.]TextControl.PageMarginB [= *margin*]

Remarks: The maximum value depends on the setting of the PageHeight Property.

Data Type: Long.

See also: PageHeight Property, PageMarginL Property, PrintDevice Property, PrintPage Property.

Example: See PrintPage Property example.

PageMarginL Property

Description: Specifies the left margin on the printed page.

Usage: [form.]TextControl.PageMarginL [= *margin*]

Remarks: The maximum value depends on the setting of the PageWidth Property.

Data Type: Long.

See also: PageHeight Property, PageWidth Property, PageMarginR Property, PrintDevice Property, PrintPage Property.

Example: See PrintPage Property example.

PageMarginR Property

Description: Specifies the right margin on the printed page.

Usage: [form.]TextControl.PageMarginR [= *margin*]

Remarks: The maximum value depends on the setting of the PageWidth Property.

Data Type: Long.

See also: PageHeight Property, PageWidth Property, PageMarginT Property, PrintDevice Property, PrintPage Property.

Example: See PrintPage Property example.

PageMarginT Property

Description: Specifies the top margin on the printed page.

Usage: [form.]TextControl.PageMarginT [= *margin*]

Remarks: The maximum value depends on the setting of the PageHeight Property.

Data Type: Long.

See also: PageHeight Property, PageMarginL Property, PrintDevice Property, PrintPage Property.

Example: See PrintPage Property example.

PageWidth Property

Description: Specifies the width of the printed page.

Usage: [form.]TextControl.PageWidth [= *height*]

Remarks: The width of the actual printed area is PageWidth minus PageMarginR minus PageMarginL. The maximum value depends on the capabilities of the selected printer and must not exceed 32767 twips. (Twips is the default scale in Visual Basic. There are 1,440 twips to one inch).

If PageWidth is 0, the Width Property is used instead. This setting can be used to place several controls without scrollbars on a page. The PageMarginL Property then determines the horizontal position of the control.

Data Type: Long.

See also: PageHeight Property, PageMarginx Properties, PrintDevice Property, PrintPage Property

Example: See PrintPage Property example.

ParagraphDialog Property

Description: Invokes the Text-Control's built-in paragraph attributes dialog box and, after the user has closed the dialog box, specifies whether he has changed something. Not available at design time; read-only at run time.

Usage: [form.]TextControl.ParagraphDialog.

Remarks: The changes made in the dialog box apply to the currently selected text. The Property settings are:

Setting	Description
True	The user has changed one or more attibutes.
False	The formatting remains unchanged.

Data Type: Boolean.

PosChange Event

Description: Occurs when the current character input position has been changed.

Usage: Sub TextControl_PosChange()

PrintDevice Property

Description: Specifies the printer device context for TextContol's built-in print function. Not available at design time.

Usage: [form.]TextControl.PrintDevice [= *device context handle*]

Data Type: Integer.

See also: PageHeight Property, PageMarginx Properties, PageWidth Property, PrintPage Property.

Example: See PrintPage Property.

PrintPage Property

Description: Prints a page of text on the default printer. Not available at design time.

Usage: [form.]TextControl.PrintPage = *page number*

Remarks: Prior to using this Property the Text-Control's output device must be selected using the PrintDevice Property.

Data Type: Integer.

See also: PageHeight Property, PageMarginx Properties, PageWidth Property, PrintDevice Property.

Example: This example shows how to print the contents of a Text-Control on the default printer:

```
Sub mnuFile_Print_Click ()
  Dim wPages, No

  Printer.Print
  wPages = TextControl1.CurrentPages
  For No = 1 To wPages
     TextControl1.PrintDevice = Printer.hDC
     TextControl1.PrintPage = No
     Printer.NewPage
  Next No
  Printer.EndDoc
End Sub
```

ReadOnly Property

Description: Specifies whether or not the Text-Control operates in read-only mode. Not available at design time.

Usage: [form.]TextControl.ReadOnly [= *boolean*]

Remarks: The ReadOnly Property settings are:

Setting	Description
True	The Text-Control operates in read-only mode.
False	The Text-Control operates in normal mode.

Data Type: Boolean.

RTFExport Property

Description: Writes the contents of a Text-Control to a file using the Rich Text Format. Not available at design time.

Usage: [form.]TextControl.RTFExport = *filename*

Remarks: RTF (Rich Text Format) is one of the most common interchange formats for text documents. Most word processors available for Windows are able to read and write RTF files.

Data Type: String.

See also: RTFImport Property.

RTFImport Property

Description: Loads the contents of an RTF file into a Text-Control. The text is inserted at the current caret position. Not available at design time.

Usage: [form.]TextControl.RTFImport = *filename*

Remarks: RTF (Rich Text Format) is one of the most common interchange formats for text documents. Most word processors available for Windows are able to read and write RTF files.

Data Type: String.

See also: RTFExport Property.

Ruler Property

Description: Specifies a ruler control to be used with a Text-Control.

Usage: [form.]TextControl.Ruler [= *ruler control name*]

Remarks: The ruler control, like the Status Bar and the Button Bar, is one of the additional controls which are contained in the tx4vb.vbx file. The chapter "Creating a Simple Word Processor" in this manual describes how to use these controls.

Data Type: String.

See also: ButtonBar Property, StatusBar Property.

Save Property

Description: Saves the contents of a text control with all its text and format information in a file. Not available at design time.

Usage: [form.]TextControl.Save = DOS File Handle

Remarks: The file must be opened in binary mode. The Visual Basic FileAttr() function is used to convert the Visual Basic file number to a DOS file handle.
The Text-Control saves its data at the current file pointer position, so it is possible to have the contents of several Text-Controls saved in one file. Also, a file header can be written by the Visual Basic program before the Save Property is used. An example of writing a file header can be found in the Text-Control MDI demo source code.

Data Type: Integer.

See also: Load Property.

Example: This example opens the file stated in the function parameter "Filename" and saves the contents of TextControl1 in it:

```
Sub SaveFileAs (Filename)
    Open Filename For Binary As #1
    TextControl1.Save = FileAttr(1, 2)
    Close #1
End Sub
```

ScrollPosX Property

Description: Specifies the position of the horizontal scrollbar. Not available at design time; read-only at run time.

Usage: [form.]TextControl.ScrollPosX

Data Type: Long.

See also: ScrollPosY Property, HScroll Event, VScroll Event.

ScrollPosY Property

Description: Specifies the position of the vertical scrollbar. Not available at design time; read-only at run time.

Usage: [form.]TextControl.ScrollPosY

Data Type: Long.

See also: ScrollPosX Property, HScroll Event, VScroll Event.

Size Event

Description: Occurs when a Text-Control has been resized with the mouse while depressing the ALT key.

Usage: Sub TextControl_Size

See also: Move Event, SizeMode Property

SizeMode Property

Description: Specifies whether the Text-Control window can be moved or resized at runtime, in the way it can at design time. If the Moveable option is selected, the control can be moved on the background by depressing the ALT key and then dragging the control with the mouse. If the Sizeable option is selected and the ALT key is depressed, the borders of the control can be dragged.

Usage: [form.]TextControl.SizeMode = *mode*

Remarks: The Property settings are:

Setting	Description
0 - Fixed	The Text-Control window cannot be moved or resized. (Default).
1 - Moveable	The Text-Control window can be moved.
2 - Sizeable	Text-Control window can be resized.
3 - Move and Sizeable	Both 1 and 2.

Data Type: Integer.

StatusBar Property

Description: Specifies the Status Bar Control to be used with a Text-Control.

Usage: [form.]TextControl.StatusBar [= *status bar control name*]

Remarks: The Status Bar control, like the Ruler and the Button Bar, is one of the additional controls which are contained in the tx4vb.vbx file. The chapter "Creating a Simple Word Processor" in this manual describes how to use these controls.

Data Type: String.

See also: ButtonBar Property, Ruler Property.

TabKey Property

Description: Determines if the Tab key is used to move the focus to the next control or to insert Tabs in the Text-Control which currently has the focus.

Usage: [form.]TextControl.TabKey [= *boolean*]

Remarks: Valid settings are:

Setting	Description
True	Inserts a Tab in the Text-Control. (Default)
False	The focus is moved to the next control.

Data Type: Boolean.

TextColor Property

Description: Determines the foreground color for selected text.

Usage: [form.]TextControl.TextColor [= *RGB value*]

Remarks: The TextColor Property applies only to the currently selected text.
Text-Control sets the value of the TextColor Property to -1 if characters with different colors have been selected. The ForeColor standard Property can be used to set the color for all the text in a control.

Data Type: Long.

VScroll Event

Description: Occurs when the vertical scroll position has been changed.

Usage: Sub TextControl_VScroll()

See also: HScroll Event.

VTSpellCheck Property

Description: Starts the spellchecker. This Property is only available if the VT-Speller tool from VisualTools has been installed. VT Speller is not part of the Text-Control package.Not available at design time; write-only at run time.

Usage: [form.]TextControl.VTSpellCheck [= *1*]

Data Type: Integer.

See also: VTSpellDictionary Property.

VTSpellDictionary Property

Description: Determines the dictionary which is used by VT-Speller.Text-Control uses this Property only if the VT-Speller tool from VisualTools has been installed. VT Speller is not part of the Text-Control package.

Usage: [form.]TextControl.VTSpellDictionary [= *filename*]

Data Type: String.

See also: VTSpellCheck Property.

Text-Control Error Codes

Two kinds of errors can occur in a Text-Control based application:

- Errors which are directly caused by using one of Text-Control's Properties. These errors can be trapped with the On Error statement. For example, setting the PageWidth to a value smaller than the right and left page margin will cause an ERR_SMALLWIDTH error.

- Errors which result from insufficient memory, corrupted files or other causes which occur within Text-Control itself. For these errors, the program receives an ErrorCode event with the error number as a parameter.

Trappable Errors

The following list describes the errors which can be trapped with the On Error statement:

Error Name	Number	Description
ERR_SMALLWIDTH	20000	Page width too small.
ERR_LARGEWIDTH	20001	Page width too large.
ERR_SMALLHEIGHT	20002	Page height too small.
ERR_LARGEHEIGHT	20003	Page height too large.
ERR_LEFTMARGIN	20004	Left margin too large.
ERR_RIGHTMARGIN	20005	Right margin too large.
ERR_TOPMARGIN	20006	Top margin too large.
ERR_BOTTOMMARGIN	20007	Bottom margin too large.
ERR_EVENT	20008	Text-Control sends an error event to specify what kind of error has occured.
ERR_WINTOOSMALL	20009	The window is too small to load the requested data.
ERR_PRINT	20010	Failure of page print.
ERR_OPENFILE	20011	OpenFile() failed.
ERR_NOLOCALMEM	20018	Out of string space.

Errors Reported by the ErrorCode Event

The following list describes the error codes which are sent by Text-Control as a parameter of the ErrorCode Event.

Error Name	No.	Description
EVERR_GLOBALMEM	1	Insufficient global memory.
EVERR_LOCALMEM	2	Insufficient local memory.
EVERR_INTERNAL	3	Internal TX error.
EVERR_FILE	4	File read/write error.
EVERR_64K_TEXT	5	Item larger than 64 KB.
EVERR_CLIPBOARD	6	Clipboard read/write error.
EVERR_MODULE	7	Module not found.

EVERR_FORMAT	8	Unknown format.
EVERR_TXT_FORMAT	9	Text filter: Unknown format.
EVERR_TXT_TOKEN	10	Text filter: Illegal token.
EVERR_TXT_READ	11	Text filter: File read error.
EVERR_TXT_WRITE	12	Text filter: File write error.
EVERR_TXT_OPEN	13	Text filter: File cannot be opened.
EVERR_TXT_SIZE	14	Text filter: File contents too large.
EVERR_TXT_UNSUPPORTED	15	Text filter: Unsupported format.

Mouse and Keyboard Assignment

Mouse Assignment

Mouse Action	Reaction of Text-Control
Click	Moves cursor to point of click or selects an image.
Shift+Click	Extends the selection to the point of click.
Double-click	Selects the word that is clicked on or opens a modal dialog box to select an image alignment.
Drag	Selects text from point of button down to point where button is released.
Double-click and drag	Extends the selection from word to word.
Triple-click and drag	Extends the selection from row to row.
PgUp/PgDown	Scrolls the text up or down one client area height minus the height of one line of text. Active only if a vertical scrollbar exists.

Moving the caret while SHIFT is pressed extends the current selection to the new caret position.

Keyboard Assignment

Key type	Reaction of Text-Control
HOME	Moves the caret to the beginning of the line.
END	Moves the caret to the end of the line.
(Left Arrow)	Moves the caret one character to the left.
(Right Arrow)	Moves the caret one character to the right.
(Up Arrow)	Moves the caret one line up.
(Down Arrow)	Moves the caret one line down.
CTRL+(Left Arrow)	Moves the caret to the beginning of the current word.
CTRL+(Right Arrow)	Moves the caret to the beginning of the next word.
CTRL+HOME	Moves the caret to start of text.
CTRL+END	Moves the caret to end of text.
CTRL+ENTER	Inserts a new page.
SHIFT+ENTER	Creates a line feed.
CTRL+(-)	Inserts an end-of-line hyphen.
DEL	Deletes selected text.
SHIFT+DEL	Copies selected text to the Clipboard and deletes the selection.
CTRL+INS	Copies selected text to the clipboard.
SHIFT+INS	Inserts text from the clipboard.
CTRL+SHIFT+(Spacebar)	Inserts a non-breaking space.
CTRL+(Backspace)	Deletes the previous word.

Moving the caret while SHIFT is pressed extends the current selection to the new caret position.

Creating a Simple Word Processor

This chapter shows you how to create a small word processor from scratch with just a few lines of code. It will be able to load and save files, use the clipboard, and will have dialog boxes for character and paragraph formatting, a ruler, a status bar and full keyboard and mouse interface.

The source code for this example is contained in the Simple sample source directory.

Creating the Project

Assuming that you have already run the Text-Control installation program and started Visual Basic, the next step is to create a project for the text processor. To do this begin by selecting the "New Project" command from the file menu. Then use the "Add File..." command to include the file tx4vbl.vbx into the new project. You will see four additional icons appear at the bottom of the Toolbox, representing the Text-Control and its Status Bar, Button Bar and Ruler:

 The Text-Control Icon

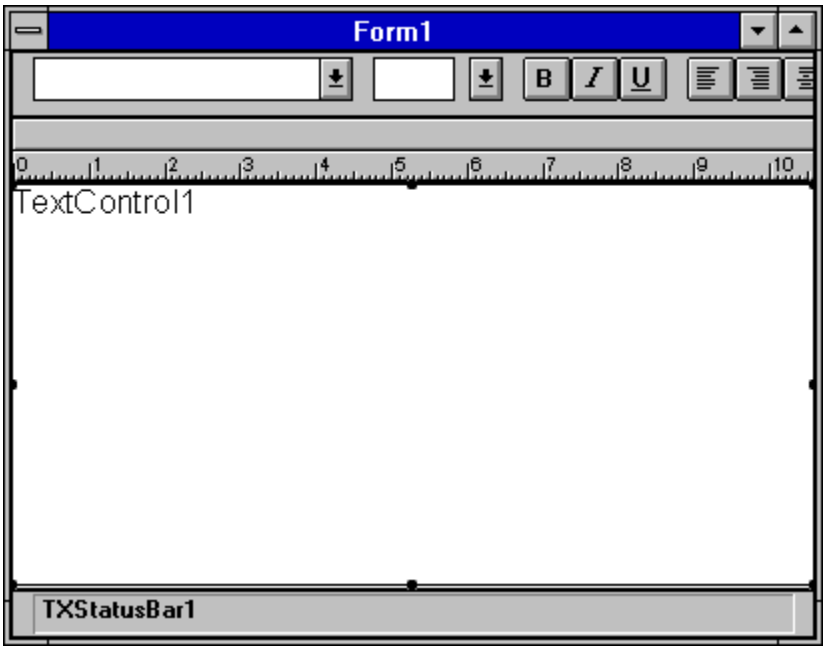
 The Status Bar Icon

 The Button Bar Icon

 The Ruler Icon

Creating the Controls

The next step is to put these four controls on a form and connect them. Click on the Text-Control icon and draw it on the form. In the same way, create a Ruler and a Button Bar on top of the Text-Control, and a Status Bar below it. Your form should now look like this:



Connecting the Controls

Click on the Text-Control to have its properties displayed. In the Property Window, look for the Properties which are named Ruler, StatusBar, and ButtonBar. Double-click on all three. Text-Control searches through the window list for its tools and enters the appropriate names.

Running the Program

The text processor is not yet finished, but we can make a first attempt at running it and seeing what it can do. Click the "Start" button. You can type in some text, select it with the mouse, copy it to the clipboard (use the <CTRL>+<INS> keys as long as there is no menu), select a different font, set tabs and do lots of other things. All of these features have been built into the Text-Control and can be used with almost no programming effort.

You will have noticed, however, that some features are still missing. For instance, if you resize the main window, the Controls keep their old sizes. There is no menu, and there are no scrollbars either. We will fix this in the coming chapters.

Adding Scrollbars

To add Scroll Bars, click on the Text-Control window to have its property list displayed. Click on the "Scrollbars" property and select "3 - Both". Select the "PageWidth" property and enter 12000, which is about the width of a letter in twips, the currently selected measurement. Set "PageHeight" to 15000 for now.

Resizing the Controls

Two steps are involved in making the controls resize properly when the main window is resized.

- Set the "Align" Property to "1 - Align Top" for the Button Bar, the Ruler and the Text-Control. Set it to "2 - Align Bottom" for the Status Bar. This will adjust everything except the height of the Text-Control.

- Open the code window for the form which contains the Text-Control. In the combo boxes on top of the code window, select "Form" in the "Object:" box and "Resize" in the "Proc:" box. The code window should show an empty procedure for the "Resize" event:

```
Sub Form_Resize ()
End Sub
```

Extend it as follows:

```
Sub Form_Resize ()
  TextControl1.Height = ScaleHeight - TXRuler1.Height -
TXStatusBar1.Height - TXButtonBar1.Height
End Sub
```

This line of code will cause the Text-Control's height to be adjusted every time the size of the form is altered.

Adding a Menu

In this chapter, you will add a menu to the text processor to enable you to call the Text-Control's built-in dialog boxes.

Use the Visual Basic Menu Design Window to create a "Format" menu with the items "Character..." and "Paragraph...". Name the items "mnuFomat_Character" and "mnuFormat_Paragraph". (Please refer to the Visual Basic documentation if you need help with creating menus).

Add the following code to the "Click" procedures of the menu items:

```
Sub mnuFormat_Character_Click ()
  Dim bChanged As Integer
  bChanged = TextControl1.FontDialog
End Sub

Sub mnuFormat_Paragraph_Click ()
  Dim bChanged As Integer
  bChanged = TextControl1.ParagraphDialog
End Sub
```

Switch the Text-Control's FormatSelection Property to "True". In the default mode, which is "False", Text-Control works like the standard Visual Basic TextBox control, so all formatting changes would apply to the whole text. In "True" mode, only selected text is effected, which is more like what you would expect from a word processor.

Start the program again. You should be able to use the menu items to call the Font and Paragraph dialog boxes.

Now for the "Edit" menu. Again use the Menu Design Window and create an "Edit" menu containing items for "Cut", "Copy", and "Paste". The code for these menu items is:

```
Sub mnuEdit_Cut_Click ()
  TextControl1.Clip = 1
End Sub

Sub mnuEdit_Copy_Click ()
  TextControl1.Clip = 2
End Sub

Sub mnuEdit_Paste_Click ()
  TextControl1.Clip = 3
End Sub
```

Having added these menu items, you can exchange formatted text with other word processors via the clipboard.

The last menu for now shall be a simple file menu. Create a "File" menu including the items "Load..." and "Save As...". Place a common dialog box icon on the form and enter the following code, which will call the common dialog box to get a file name from the user, and will then load respectively save the selected file:

```
Sub mnuFile_Load_Click ()
  On Error Resume Next

' Create an "Open File" dialog box
  CMDialog1.Filter = "TX Demo (*.tx)|*.tx"
  CMDialog1.DialogTitle = "Open"
  CMDialog1.Flags = &H1000& ' OFN_FILEMUSTEXIST
  CMDialog1.CancelError = True
  CMDialog1.Action = 1
```

```
  If Err Then Exit Sub

  ' Open the selected file
  Open CMDialog1.Filename For Binary As #1
  If Err Then
    MsgBox "Can't open file: " + CMDialog1.Filename
    Exit Sub
  End If

  ' Use the FileAttr function to get a DOS file handle
  ' from the VisualBasic file number and pass it on to TX
  TextControl1.Load = FileAttr(1, 2)
  Close #1
  TextControl1.Refresh

End Sub

Sub mnuFile_SaveAs_Click ()
  On Error Resume Next

  ' Create a "Save File" dialog box
  CMDialog1.Filter = "TX Demo (*.tx)|*.tx"
  CMDialog1.DialogTitle = "Save As"
  CMDialog1.Flags = &H2& ' OFN_OVERWRITEPROMPT
  CMDialog1.CancelError = True
  CMDialog1.Action = 2

  If Err Then Exit Sub

  ' Open the selected file
  Open CMDialog1.Filename For Binary As #1
  If Err Then
    MsgBox "Can't open file: " + CMDialog1.Filename
    Exit Sub
  End If

  TextControl1.Save = FileAttr(1, 2)
  Close #1

End Sub
```

What Comes Next

Text-Control has of course many more features than those included in our little demo program. It is up to you now to include zoom, paragraph frames and whatever else makes up a full-blown word processor. If you need some hints about how to integrate special features, have a look at the source code of the other sample programs.

Text-Control Programming

This chapter is a guide to programming Text-Control and its tools, explaining the parts which have been omitted from the "Creating a Simple Word Processor" example.

Working with Files

Text-Control uses two different file formats:

- Its own, native format, which you would normally use to store data in document files.

- The Rich Text Format (RTF), which can be used to exchange formatted text with other applications.

An example of how to use the native file format has already been presented in the previous chapter. Using RTF is even simpler: Just assign a file name to the RTFImport or RTFExport Property to load or save a file.

Flexibility is the reason why using the native file format has to be a bit more complicated. With the RTF Properties, you can only read or write the contents of a single Text-Control from or to a file. Using the Load and Save Properties, you can write a file header prior to saving the Text-Control data, or even write the contents of several Text-Controls to one file. The Forms1 sample program, which is described in the next but one chapter, shows you how to do this.

Printing

Visual Basic provides two techniques for sending information to the printer. The first one is to use the PrintForm Method, the second is to use the printer object. Both Methods have their drawbacks: PrintForm works with screen resolution only, which would result in very poor print quality. The printer object, on the other hand, provides the best print quality, but requires a lot of coding. Text-Control uses the second method to achieve the best result, but without the "lot of coding".

The following example sends the contents of a Text-Control, which can be several pages long, to the default printer:

```
Sub mnuFile_Print_Click ()
  Dim wPages, No

  Printer.Print
  wPages = TextControl1.CurrentPages
  For No = 1 To wPages
     TextControl1.PrintDevice = Printer.hDC
     TextControl1.PrintPage = No
     Printer.NewPage
  Next No
  Printer.EndDoc
End Sub
```

After initializing the printer object with the "Printer.Print" statement, the number of pages is stored in a local variable called "wPages". The following "For .. Next" loop runs from 1 to "wPages" to print all of the pages. Inside the loop there are three lines of code which print a single page:

1. The device context handle of the printer object is assigned to Text-Control's PrintDevice Property. Without this step, a device context which is compatible to the screen device would be used, resulting again in poor print quality.

2. The number of the page to be printed is assigned to the PrintPage Property. This will also start the printing process.

3. The Printer object's NewPage method is invoked to advance to the next page.

Everything else, like calculating the line and page breaks, is done internally by Text-Control. The formatting is based on the values of two groups of Properties:

- PageHeight and PageWidth determine the dimensions of the printed page.

- PageMarginB, PageMarginL, PageMarginR andPageMarginT determine the print margins.

These Properties are normally set in a print dialog box.

A Word Processor

This chapter shows you how to use Text-Control to write a standard word processor. The program is based upon the MDI sample from the Visual Basic Programmer's Guide, with the TextBox controls replaced by Text-Controls. If you are not familiar with MDI, control arrays or creating a toolbar you may want to read that chapter first.

The source code for this example is contained in the MDIDEMO sample source directory.

Adding a PageSetup Dialog Box

The Page Setup dialog, box is used to determine the page size and print margins. The maximum page size is resticted by the capabilities of the default printer. For implementation details, look at the source code of the DOCDLG form.

A Print Dialog Box

When the "Print..." menu item is clicked, first a Common Dialog box is shown to let the user enter the range of pages, number of copies and printer specific information. The rest of the procedure, which is part of the MDIChild form, is just a loop which for every page to be printed sets the appropriate Text-Control properties.

Search and Replace

Searching and replacing is entirely done in Text-Control. You just have to assign a value of 1 for Search or 2 for Search And Replace to the FindReplace property. The VBX then opens the Windows Common Dialog box.

Dialog Boxes for Text and Background Color

This is also done with Common Dialogs. The color value returned from the dialog box is assigned to the TextColor or BackColor properties.

Using Paragraph Frames

With Text-Control, you can add lines and frames to a paragraph or a range of paragraphs. For instance, you can put a line on top of a caption like in the Property reference of this manual. Or you can create tables by using the "tab lines" feature which draws a vertical line at every tab stop.

The dialog box for paragraph frames is not included in the Text-Control, but the source code is included in the MDI sample.

The Propeties which are responsible for paragraph frames are FrameDistance, FrameLineWidth, and FrameStyle.

Adding a Spell Checking Tool

Text-Control has no built-in spell checker, but can be used with the VT-Speller tool from VisualTools, Inc. Having installed VT-Speller, all you have to do is to start it with just one line of Basic code which assigns a value of 1 to Text-Control's VTSpellCheck action property:

```
VTSpellCheck = 1
```

It is not necessary to put a VT-Speller icon on the form or to add it to your project.

You can start the spellchecker, for instance, from a toolbar button or from a menu item. The spellchecking process is handled entirely by VT-Speller's built-in dialog boxes.

Another Property, VTSpellDictionary, enables you to specify a different dictionary for the spellchecker. Dictionaries can be created and edited with a tool which is part of the VT-Speller package.

Using Text-Control with C++

The Text-Control VBX is compatible with Visual Basic 1.0. This enables you to use it with any compiler which supports VBX controls, like Microsoft Visual C++, Borland C++, dBase for Windows and others. Sample programs for the Microsoft and Borland C++ compilers are to be found in the \samples\mfc and \samples\owl subdirectories.

Please note that C++ compilers use different coordinate systems to Visual Basic. The values of the Left, Top, Width, Height, PageWidth, and PageHeight Properties, for instance, are always specified in pixels. Please refer to the sample programs to see how to use these properties.