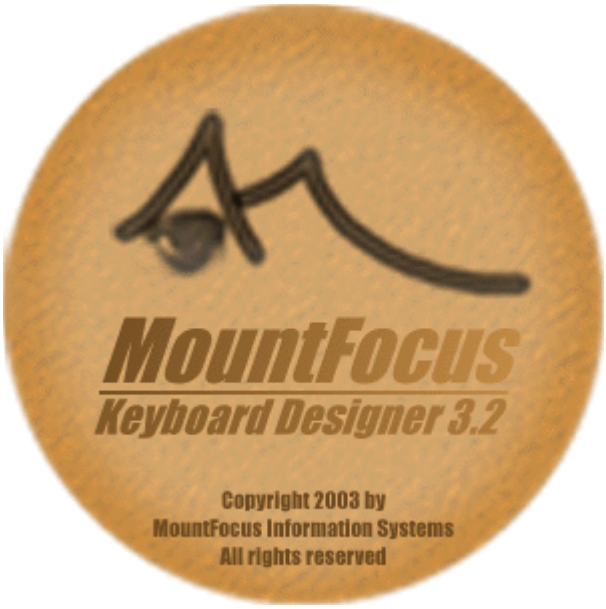


MountFocus Keyboard Designer Manual

Table of Contents

Part I Overview 8

Part II What's New in Version 3.2? 10

Part III Quick Start Guide 12

1 Introduction ... 12
- What is the MountFocus Keyboard Designer? ... 12
- Documentation ... 12

2 Keyboard Basics ... 12
- A Tour of the Keyboard Designer ... 12
  - Starting the Keyboard Designer ... 12
  - Using toolbars, menus, and keyboard shortcuts ... 14
  - Creating a keyboard ... 15
  - Placing and moving objects on a keyboard ... 15
  - Changing objects appearance and behavior ... 16
    - Working with the property editor ... 16
    - Working with the object list ... 17

3 Starting Out ... 17
- Your first keyboard - a short tutorial ... 17
  - Creating a new keyboard ... 18
  - Setting keyboard values ... 18
  - Adding a key to the keyboard ... 19
  - Adding a caption to a key ... 21
  - Adding functionality to your key ... 23
  - Adding additional objects ... 26
  - Testing your Keyboard ... 26

4 Nice to Know ... 27
- Customizing the Keyboard Designer ... 27
- Organizing your work area ... 27
  - Arranging windows ... 27
  - Setting Defaults ... 27

Part IV Keyboard Designer Manual 29

1 How To 29
- Set defaults ... 29
- Identify objects ... 29
- Place objects on a keyboard ... 29
- Size objects ... 30
- Move objects ... 31
- Create bevel effects ... 31

2 Objects Explained ... 32
- Keyboard ... 32
- Key ... 32
- Label ... 32

© 2002 MountFocus Information Systems

Image 33
Indicator 33
Region 33
Panel 33
Page control 34
Page 34

3 The Keyboard Designer Interface 34

Menu Items 35
File Menu 36
Edit Menu 36
View Menu 37
Tools Menu 38
Window Menu 39
Help Menu 39
Right Click Menus 39
Toolbars 40
Keypool 41
Options 41
Editors 42
Property Editor 43
Events 44
Objects 44
Caption Editors 44
Indicator (State) Editor 45
Keystrokes Editor 46
Script Editor 49
Strings Editor 49
Scale 50

Part V Runtime Keyboard Control Panel 52

Part VI Runtime Keyboard reference 57

1 Runtime Keyboard features 57
2 Command line parameters 59
3 System tray menu 60
4 Using COM 60

COM properties 62
COMActivePageIndex 62
COMActivePageName 62
COMBlend 63
COMDelay 63
COMHeight 63
COMKeepOnDesktop 63
COMKeyboardAsBinary 64
COMKeyboardAsText 64
COMKeyboardFile 64
COMLeft 64
COMMenuOn 65
COMMoveMode 65
COMPageControlPageCount 65
COMPageControlPageIndex 65
COMPageControlPageName 66

© 2002 MountFocus Information Systems

COMPageCount ... 66
COMPosition ... 66
COMSoundOn ... 67
COMTop ... 67
COMTrNumPad ... 67
COMVisible ... 68
COMWait ... 68
COMWidth ... 68
COMWinHandle ... 68
COM methods ... 69
COMClosePage ... 69
COMHide ... 69
COMLoadKeyboard ... 69
COMPageControlClose ... 69
COMSetIndicator ... 70
COMSetToTop ... 70
COMShow ... 70
5 Writing scripts ... 70
Kbd Object ... 71
Kbd Object methods ... 71
Kbd.GetControl ... 71
Kbd.ShellExecute ... 71
Part VII Classes 74
1 Surface classes ... 74
TmfKBSurface ... 74
TmfKBSurfaceProperties ... 74
TmfKBSurface.Bevel ... 74
TmfKBSurface.Color ... 74
TmfKBSurface.pLeft ... 75
TmfKBSurface.pTop ... 76
TmfKBSurface.sHeight ... 76
TmfKBSurface.sWidth ... 76
TmfKBSurfaceEvents ... 76
TmfKBSurface.OnButtonDown ... 76
TmfKBSurface.OnButtonUp ... 77
TmfKBSurface.OnMouseEnter ... 77
TmfKBSurface.OnMouseLeave ... 77
TmfKBSurface.OnPointerMove ... 78
TmfKBPage ... 78
TmfKBPageProperties ... 78
TmfKBPageEvents ... 78
TmfKBPage.OnActivate ... 78
TmfKBPage.OnDeactivate ... 79
TmfKBPageMethods ... 79
TmfKBPage.Activate ... 79
TmfKBSizeableSurface ... 79
TmfKBSizeableSurfaceProperites ... 79
TmfKBKeyboard ... 80
TmfKBKeyboardProperties ... 80
TmfKBKeyboard.GlobalPageControl ... 80
TmfKBKeyboard.GlobalPageCount ... 80
TmfKBKeyboard.GlobalPageIndex ... 80

© 2002 MountFocus Information Systems

TmfKBKeyboard.GlobalPageName ... 81
TmfKBKeyboard.GridHeight ... 81
TmfKBKeyboard.GridWidth ... 81
TmfKBKeyboard.IndStates ... 81
TmfKBKeyboard.KeySound ... 82
TmfKBKeyboard.ScriptLanguage ... 82
TmfKBKeyboard.TimerEnabled ... 82
TmfKBKeyboard.TimerInterval ... 82
TmfKBKeyboardEvents ... 83
TmfKBKeyboard.OnHide ... 83
TmfKBKeyboard.OnLoad ... 83
TmfKBKeyboard.OnShow ... 83
TmfKBKeyboard.OnTimer ... 84
TmfKBKeyboardMethods ... 84
TmfKBKeyboard.GetControl ... 84
TmfKBKeyboard.GetIndState ... 84
TmfKBMoveableSurface ... 85
TmfKBMoveableSurfaceProperties ... 85
TmfKBPanel ... 85
TmfKBPanelProperties ... 85
TmfKBPanelEvents ... 86
TmfKBPageControl ... 86
TmfKBPageControlProperties ... 86
TmfKBPageControl.CurrentPage ... 86
TmfKBPageControl.DefaultPage ... 87
TmfKBPageControl.PageCount ... 87
TmfKBPageControl.PageIndex ... 87
TmfKBPageControl.PageName ... 87
TmfKBPageControlEvents ... 87
TmfKBPageControl.OnPageChanged ... 88
TmfKBPageControl.OnPageChanging ... 88
TmfKBPageControlMethods ... 88
TmfKBPageControl.ClosePage ... 88

2 Control classes ... 89

TmfKBControl ... 89
TmfKBControlProperties ... 89
TmfKBControl.Color ... 89
TmfKBControl.pLeft ... 90
TmfKBControl.pTop ... 90
TmfKBControl.sHeight ... 90
TmfKBControl.sWidth ... 90
TmfKBControl.Transparent ... 90
TmfKBControlEvents ... 91
TmfKBControl.OnButtonDown ... 91
TmfKBControl.OnButtonUp ... 91
TmfKBControl.OnMouseEnter ... 92
TmfKBControl.OnMouseLeave ... 92
TmfKBControl.OnPointerMove ... 92
TmfKBKey ... 93
TmfKBKeyProperties ... 93
TmfKBKey.Background ... 93
TmfKBKey.Caption ... 94
TmfKBKey.CloseKeyboard ... 94
TmfKBKey.ClosePage ... 94

© 2002 MountFocus Information Systems

TmfKBKey.KeyState ... 94
TmfKBKey.Keystrokes ... 95
TmfKBKey.KeyType ... 95
TmfKBKey.MoveKeyboard ... 95
TmfKBKey.PressedBevel ... 96
TmfKBKey.PressedOffsetX|Y ... 96
TmfKBKey.ReleasedBevel ... 96
TmfKBKey.Repeating ... 96
TmfKBKey.SelectPage ... 98
TmfKBKey.ShiftState ... 98
TmfKBKeyEvents ... 98
TmfKBKey.OnKeyPressed ... 98
TmfKBKey.OnKeyReleased ... 99
TmfKBRegion ... 99
TmfKBRegionProperties ... 99
TmfKBStaticControl ... 99
TmfKBStaticControlProperties ... 99
TmfKBStaticControlEvents ... 100
TmfKBIndicator ... 100
TmfKBIndicatorProperties ... 100
TmfKBIndicator.MonitorIndicator ... 100
TmfKBIndicator.OffBevel ... 100
TmfKBIndicator.OffCaption ... 101
TmfKBIndicator.OnBevel ... 101
TmfKBIndicator.OnCaption ... 101
TmfKBIndicator.State ... 101
TmfKBIndicatorEvents ... 102
TmfKBIndicator.OnStateChanged ... 102
TmfKBIndicator.OnStateChanging ... 102
TmfKBLabel ... 102
TmfKBLabelProperties ... 103
TmfKBLabel.AutoSize ... 103
TmfKBLabel.Bevel ... 103
TmfKBLabel.Font ... 103
TmfKBLabel.Lines ... 103
TmfKBLabel.Text ... 103
TmfKBImage ... 104
TmfKBImageProperties ... 104
TmfKBImage.AutoSize ... 104
TmfKBImage.Bevel ... 105
TmfKBImage.Image ... 105
TmfKBImage.UseColor ... 105

3 Non visual classes ... 105

TmfKBNonVisual ... 105

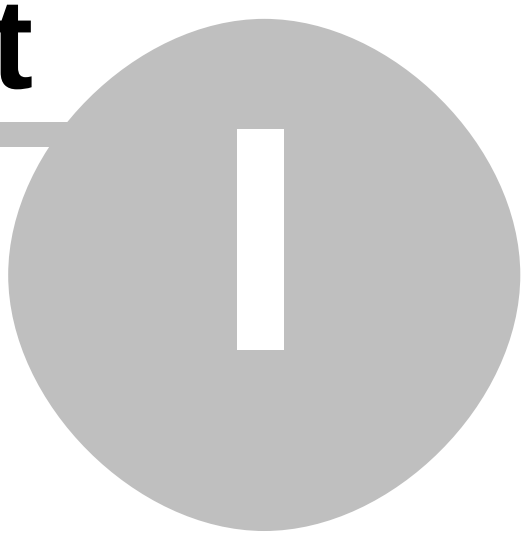
TmfKBKeystrokes ... 105
TmfKBKeyNode ... 105
TmfKBSound ... 105
TmfKBSoundMethods ... 106
TmfKBSound.LoadFromFile ... 106
TmfKBSound.Play ... 106
TmfKBBevel ... 106
TmfKBBevelProperties ... 107
TmfKBBevel.Border ... 107
TmfKBBevel.BorderWidth ... 107

© 2002 MountFocus Information Systems

TmfKBBevel.BottomColor . . . 108
TmfKBBevel.Inner . . . 108
TmfKBBevel.InnerWidth . . . 108
TmfKBBevel.Outer . . . 108
TmfKBBevel.OuterWidth . . . 108
TmfKBBevel.Spacing . . . 109
TmfKBBevel.TopColor . . . 109
TmfKBIndState . . . 109
TmfKBIndStateProperties . . . 109
TmfKBIndState.IndNo . . . 110
TmfKBIndState.Name . . . 110
TmfKBIndState.State . . . 110
TmfKBIndStateEvents . . . 110
TmfKBIndState.OnStateChanged . . . 110
TmfKBIndState.OnStateChanging . . . 111
TPicture . . . 111
TPictureMethods . . . 111
TPicture.LoadFromFile . . . 111
TFont . . . 112
TFontProperties . . . 112
TFont.Charset . . . 113
TFont.Color . . . 114
TFont.Height . . . 114
TFont.Name . . . 114
TFont.Pitch . . . 115
TFont.Size . . . 115
TFont.Style . . . 116

© 2002 MountFocus Information Systems

Part



1 Overview

Welcome to the MountFocus Keyboard Designer – the ultimate onscreen keyboard design tool.

The MountFocus Keyboard Designer is capable of producing keyboards for a wide variety of situations. There are literally hundreds of uses for this powerful onscreen virtual keyboard. Any computer using a Windows system, touch-screen or conventional, can use any keyboard designed with the MountFocus Keyboard Designer. That makes it ideal for situations where a physical keyboard is not available, not desired, or just not feasible. Some examples are:

Point of Sale applications
Infokiosks
Remote customer terminals
ATM machines
Personal computers
Computers for disabled users

Any time a conventional keyboard is not the best choice, are perfect situations for the MountFocus Keyboard Designer.

The MountFocus Keyboard Designer package also includes ready made keyboards which can be installed without having to design a keyboard from scratch and only need the Runtime Keyboard license installed to make them work.

Software developers
Keyboards designed with the MountFocus Keyboard Designer are touch-screen compatible, and can also be used on non touch-screen hardware, without wasting hours of time rewriting code.

Multiple lines of caption (text and/or graphics), background color and/or graphics, multiple keystrokes from one key, multiple "page controls" for context sensitive keyboards are all possible. You can integrate your own systems with the MountFocus Keyboard Designer using OLE. This allows your applications to change the keyboard layout at runtime by switching pages.

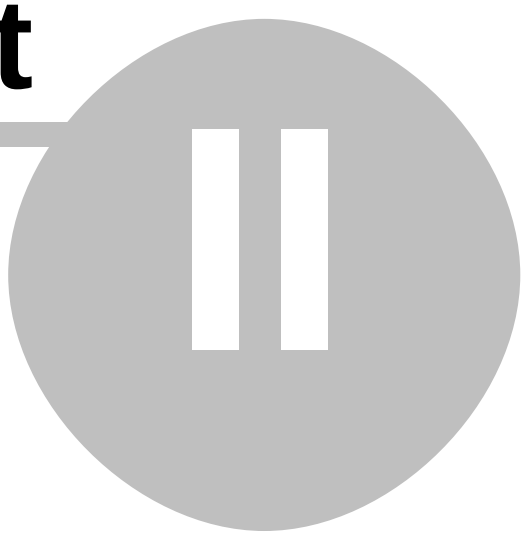
Disabled users
Do you have difficulty using a standard keyboard? Now you can have all the features of the standard computer keyboard on your screen with the click of a mouse or touch of your finger. You can start programs, enter text, create special keys for those hard to type functions - the uses are endless. Every feature of the standard keyboard is available on screen. Need a keyboard with special functions? The MountFocus Keyboard Designer is so easy to use even a novice can design the perfect keyboard for themselves. Take the tutorial, copy keys from the pool, you'll have your own custom made keyboard by lunchtime.

Touchscreen users or touchscreen software developers
Any touchscreen can be turned into a virtual keyboard with custom designed keyboards or standard layouts. Touchscreens eliminate the need for keyboards, and with the MountFocus Keyboard Designer you can "get at" the programs running on your touchscreen system. Run any Windows program on your touchscreen system with the MountFocus Keyboard Designer.

Where to go from here
Click on the Contents or Index buttons on the toolbar to search for a specific topic. Due to the large amount of changes to this version, even experienced users are encouraged to read the tutorial.

© 2002 MountFocus Information Systems

Part



2 What's New in Version 3.2?

Version 3.2 is a minor upgrade, mostly to add user requested features and bug fixes

The most important updates are:

Unicode
Full Unicode support has been added, including the following:

- Keystrokes can be marked as Unicode even if the code is less than $100.
- The extended flag cannot be set on Unicode characters.

Key Types
There are now 3 types of keys, Normal, Shift and ShiftLock.

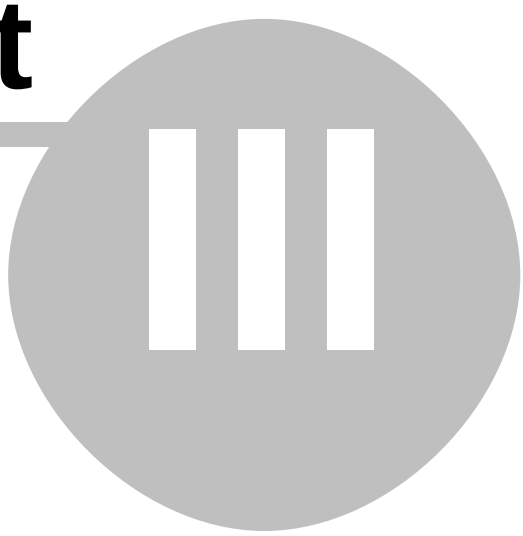
Keyboard Designer
- The Keystrokes editor has been changed slightly to allow keystrokes to be marked as Unicode and an additional Extended flag button has been added.
- You can now create Unicode labels which work on Windows 98 and above.

Runtime Keyboard
- The COM interface now publishes two new properties
- KeepOnDesktop: when set to True (or 1) behaves as before. When set to False (or 0) it allows the Runtime Keyboard to move or be dragged outside the desktop area. The default is True.
- Delay: A global delay between keystrokes can now be set specified in milliseconds.

Old version keyboards
Keyboard files are backwards compatable but Keyboard files saved in version 3.2 format may not open in earlier versions of the Keyboard Designer.

© 2002 MountFocus Information Systems

Part



3 Quick Start Guide

3.1 Introduction

This Quick Start Guide provides an overview of the MountFocus Keyboard Designer so you can get started using the software in a short amount of time. It also gives you the basic knowledge needed to create your first Keyboard project.

3.1.1 What is the MountFocus Keyboard Designer?

The MountFocus Keyboard Designer is a tool for creating virtual keyboards on the Windows platform. Keyboards created using the MountFocus Keyboard Designer can be used with any Windows application in a touch-screen or normal environment. That makes it ideal for situations where a physical keyboard is either not available or not desired.

The MountFocus Keyboard Designer was designed to allow developers or users of Windows applications to design their own on-screen virtual keyboards. The design was originally made for Point of Sale applications, but it is suitable for other uses as well. Infokiosks, remote customer terminals, or any application where you do not want the user to have a physical keyboard available, are ideal situations for the MountFocus Keyboard.

Multiple "pages" allow context sensitive keyboard layout allowing the user to press "legal" keys only. OLE support allows other applications to select pages and control the appearance of the keyboard. Easy deployment of your custom made keyboards. Single or multiple keystrokes can be sent from a single key press. Multiple objects can be placed on keys, the keyboard surface, page controls, and panels. Graphics or text can be used. Copy keys from the integrated key pool for fast and easy creation of new keyboards. The most flexible and powerful virtual keyboard designer available. Excellent for touch-screen, Point Of Sale and similar applications.

3.1.2 Documentation

Documentation for the MountFocus Keyboard Designer is available in several formats:

Adobe .PDF file
HTML Help

3.2 Keyboard Basics

3.2.1 A Tour of the Keyboard Designer

This chapter will walk you through a short introduction to the Keyboard Designer environment and help you become familiar with the various parts of the program.

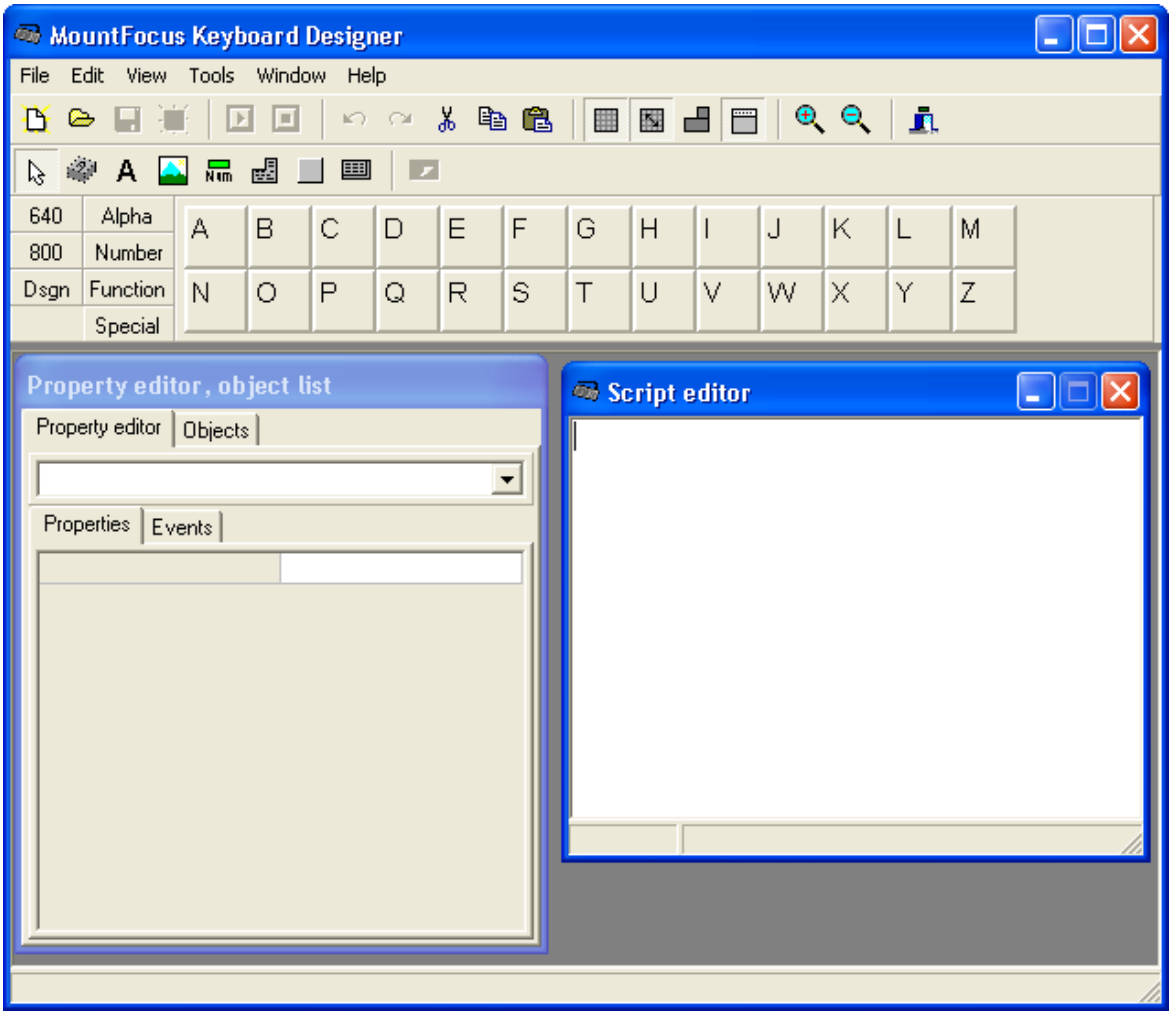
3.2.1.1 Starting the Keyboard Designer

You can start the Keyboard Designer in two ways:

- Double-click the Keyboard Designer icon from Explorer.
- Choose Programs | MountFocus POS | Keyboard Designer from the Windows Start menu.

© 2002 MountFocus Information Systems

The first thing you will see is the main screen of the Keyboard Designer with the property editor and object list open by default.



© 2002 MountFocus Information Systems

3.2.1.2 Using toolbars, menus, and keyboard shortcuts

The Keyboard Designer's toolbars, located in the main window, provide quick access to frequently used operations and commands. All toolbar operations are duplicated in the drop-down menus.

Main Toolbar

Tool	Name	Action
	New	Create new keyboard
	Open	Open keyboard file
	Save	Save keyboard file
	Save as	Save keyboard file with new name
	Run file	Run open keyboard file
	Close runtime	Close open keyboard file
	Undo	Undo last action
	Redo	Redo last action
	Cut	Cut object for pasting
	Copy	Copy object for pasting
	Paste	Paste cut or copied object
	Show grid	Toggle between show/hide grid
	Snap to grid	Line up objects on grid lines
	Show regions	Toggle between show/hide regions
	Show keypool	Toggle between show/hide keypool
	Zoom in	Increase view scale
	Zoom out	Decrease view scale
	Exit	Exit program

© 2002 MountFocus Information Systems

Object Toolbar

Tool	Name	Action
	Select	Select object
	Key	Create new key object
	Label	Create new label object
	Image	Create new image object
	Indicator	Create new indicator object
	Region	Create new region object
	Panel	Create new panel object
	Page control	Create new page control object
	Create new page	Create new page on page control object

Many operations have keyboard shortcuts as well as toolbar buttons. When a keyboard shortcut is available, it is always shown next to the command in the drop-down menu.

Positioning the cursor over an icon on the toolbar will display a short description of the action the tool will perform.

You can right-click on all objects to display a menu of commands appropriate to the object you are working with. Some commands are only available from the right-click menu.

3.2.1.3 Creating a keyboard

The first step to making an on-screen keyboard is to create a new keyboard file.

You can choose File | New from the drop-down menu, click on the New File icon on the toolbar, or press Ctrl + N.

You are now ready to start placing objects on your keyboard.

3.2.1.4 Placing and moving objects on a keyboard

New objects can be placed on the keyboard by choosing a tool from the Tool drop-down menu or by clicking on the tool icon in the toolbar and then clicking on the keyboard to place the object.

Objects can be moved by dragging or by using the arrow keys..

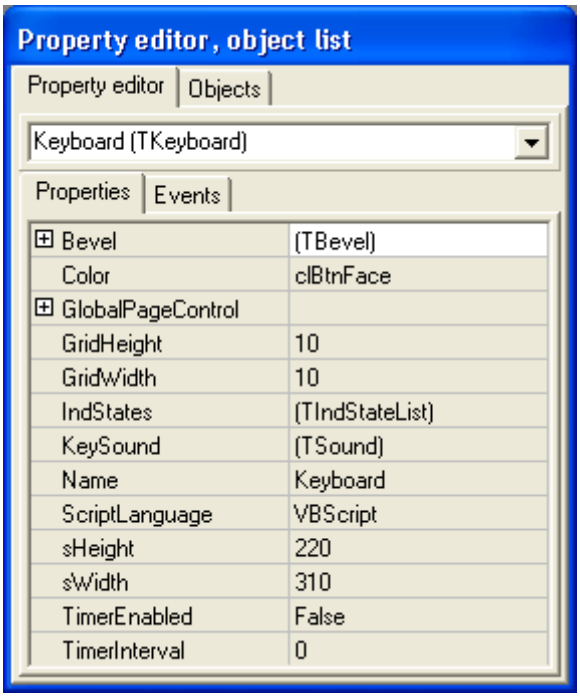
© 2002 MountFocus Information Systems

3.2.1.5 Changing objects appearance and behavior

This section will give you a brief overview of using the Property Editor and Object List.

3.2.1.5.1 Working with the property editor

To change the appearance and location of objects on the keyboard, changes are made in the property editor.

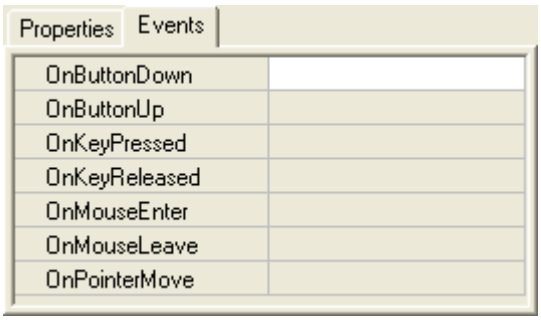


All changeable properties of the chosen object are displayed, and if more than one object is chosen, only the properties they have in common are displayed.

Some properties have simple values -- such as *True* or *False*. Others have associated property editors to set more complex values. When you click on such a property value, you'll see an ellipsis. Still others have a default value which can be changed by simply typing a new value in the box.

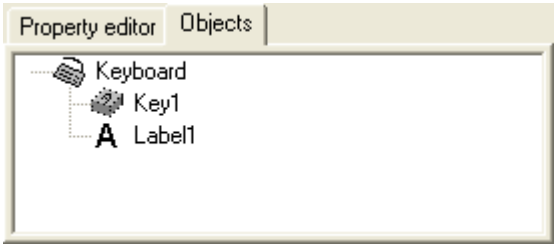
© 2002 MountFocus Information Systems

Clicking on the events tab will show a list of events which the object can handle.



3.2.1.5.2 Working with the object list

Clicking on the object list tab will show a list of objects on the keyboard in tree view. Clicking on any object will select that object.



3.3 Starting Out

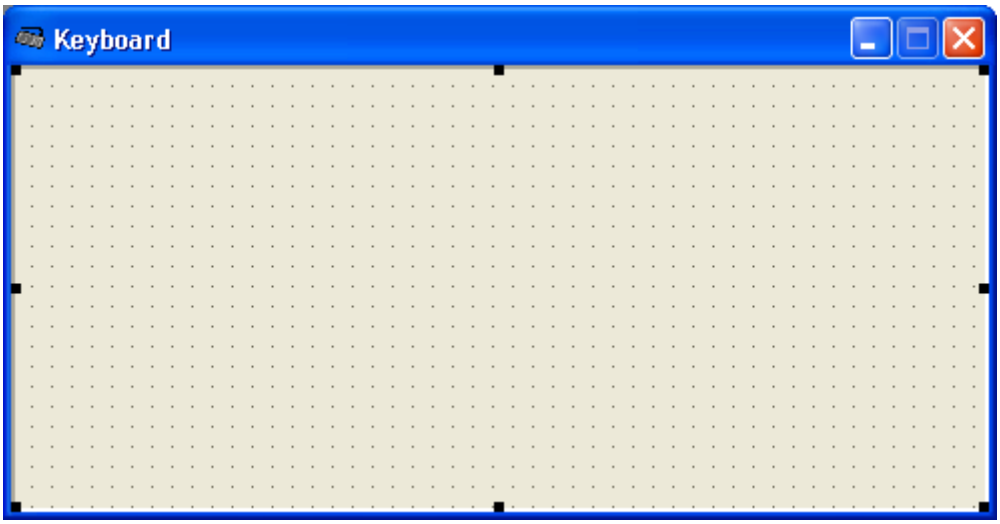
3.3.1 Your first keyboard - a short tutorial

The fastest way to introduce yourself to the Keyboard Designer is to create a keyboard. This tutorial will guide you through creating a simple one-key keyboard. You'll place an object on the keyboard and a caption on that object. Once your keyboard is complete, you'll be able to test it by running the keyboard file and using it with an application.

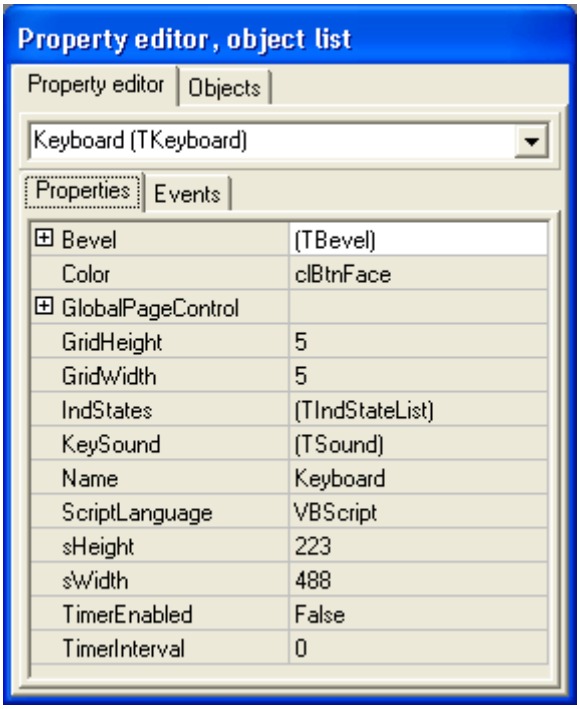
© 2002 MountFocus Information Systems

3.3.1.1 Creating a new keyboard

When you create a new keyboard, the Keyboard Designer displays a blank keyboard.



Next to the keyboard, you'll see the Property editor, which you can use to set property values for the keyboard and objects you place on it.



3.3.1.2 Setting keyboard values

When you use the Property Editor to set properties, the Keyboard Designer will show you the changes you make, after you press *Enter*.

- Set the color of the keyboard to Sky Blue.

© 2002 MountFocus Information Systems

Find the keyboard's Color property in the Property Editor and click the drop-down list displayed to the right of the property. Choose *clSkyBlue* from the list.

There are several other properties which can be set, but we won't change those for the moment.

3.3.1.3 Adding a key to the keyboard

The objects toolbar has several icons representing the different objects which can be placed on the keyboard. Add an object to the keyboard by selecting the object on the toolbar, then clicking on the keyboard where you want to place it. You can also double-click on an icon to place more than one instance of that object by clicking repeatedly on the keyboard.

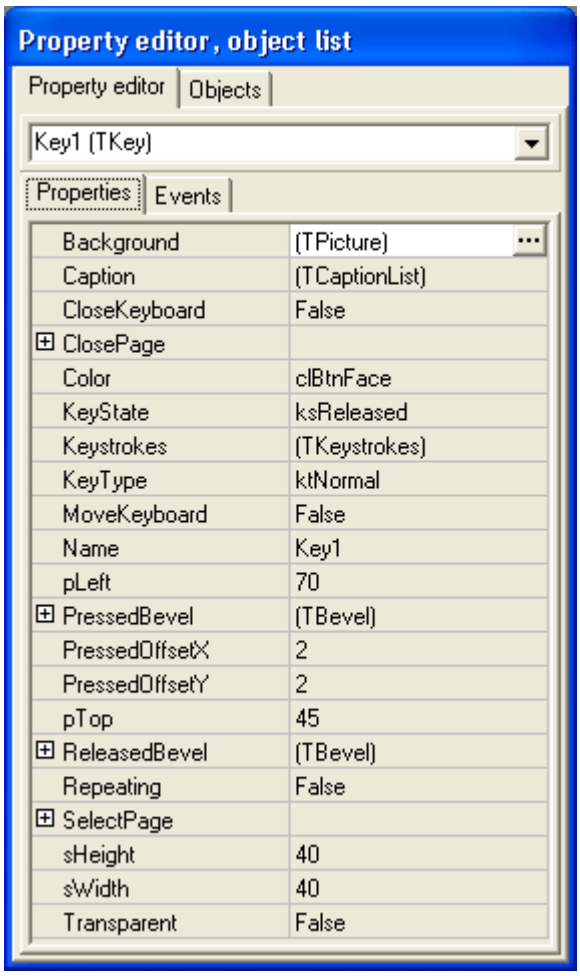
You'll notice that the cursor changes appearance according to how the object icon was clicked. If the icon was clicked once, the cursor appears white with black text. If the icon was double-clicked, the cursor appears black with white text. The cursor will also change appearance according to which icon was clicked.

Add a key to the keyboard:

- Click on the key tool and again on the keyboard where you would like to place the key. For this tutorial, let's place the key in the upper left corner of the keyboard.

© 2002 MountFocus Information Systems

Once you have placed a key on the keyboard, the Property Editor will change to show the properties of the key. This is where you can change the appearance of the key, add captions, images, etc.



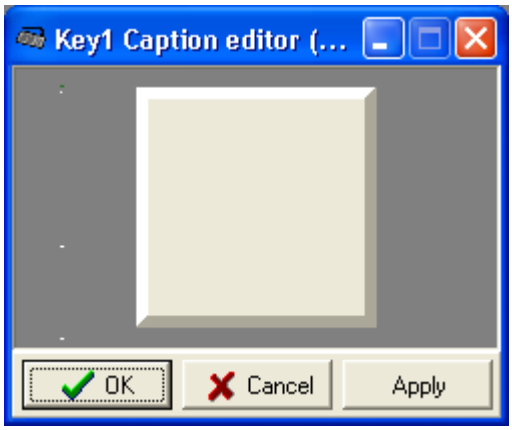
For this tutorial, we'll just add a simple caption.

© 2002 MountFocus Information Systems

3.3.1.4 Adding a caption to a key

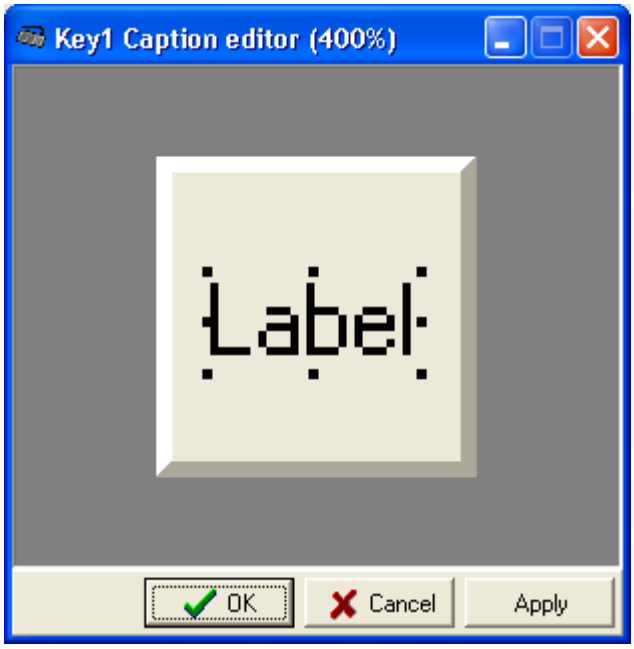
1. Find the key's Caption property and click on the property value to the right to make the ellipsis button appear. Click again on the ellipsis button to show the Caption Editor.

   You should now have a new window with a zoomed in image of the key, and you can place something on the surface of the key. The objects which can be placed on a key are: Labels, Images, and Indicators. Notice that the Property Editor shows no properties because we haven't placed any objects on the key yet.



   Let's add a label to the key.

2. Click the Label tool and click again on the key in the Caption Editor.

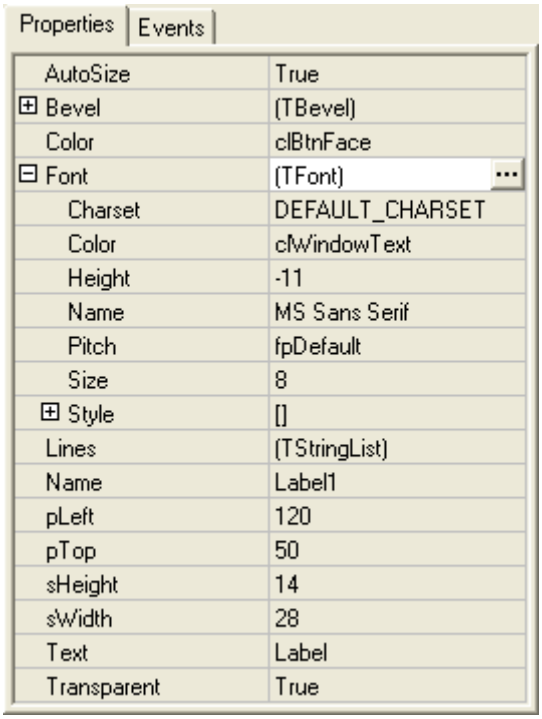


3. Find the Label's Text property and click on the word "Label" to change the text. Now type `Hi!` and press *Enter*.

© 2002 MountFocus Information Systems

You'll notice that the text doesn't quite fit on the key.

4. Let's change the font. Find the Font property in the Property Editor and click on the "+" sign to the left. This will expand the available sub-properties to show more options.



Find the Name property under Font and choose Arial from the drop-down list and press *Enter*.

5. Find the Size property and change the value to 7 and press *Enter*.

That looks better!

6. Now you can click on Apply or OK and the changes will be shown on the key on the keyboard.

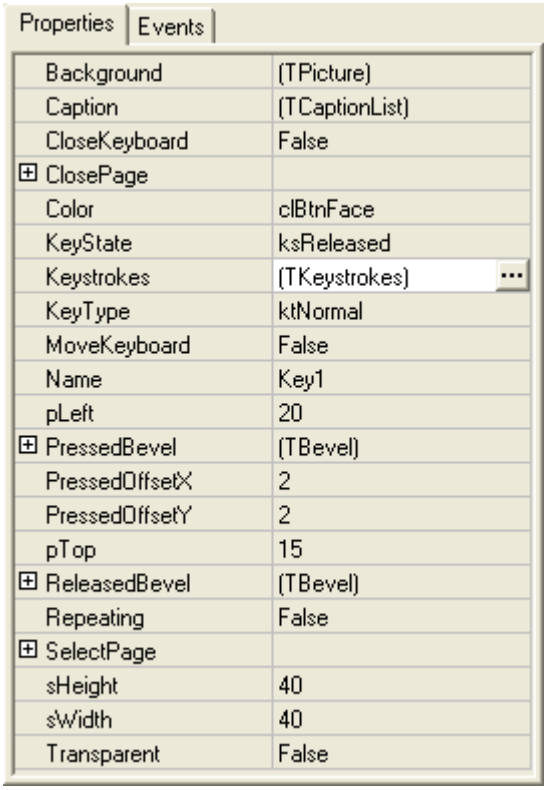
© 2002 MountFocus Information Systems

3.3.1.5 Adding functionality to your key

Now that you have a key, you'll need to make that key actually do something.

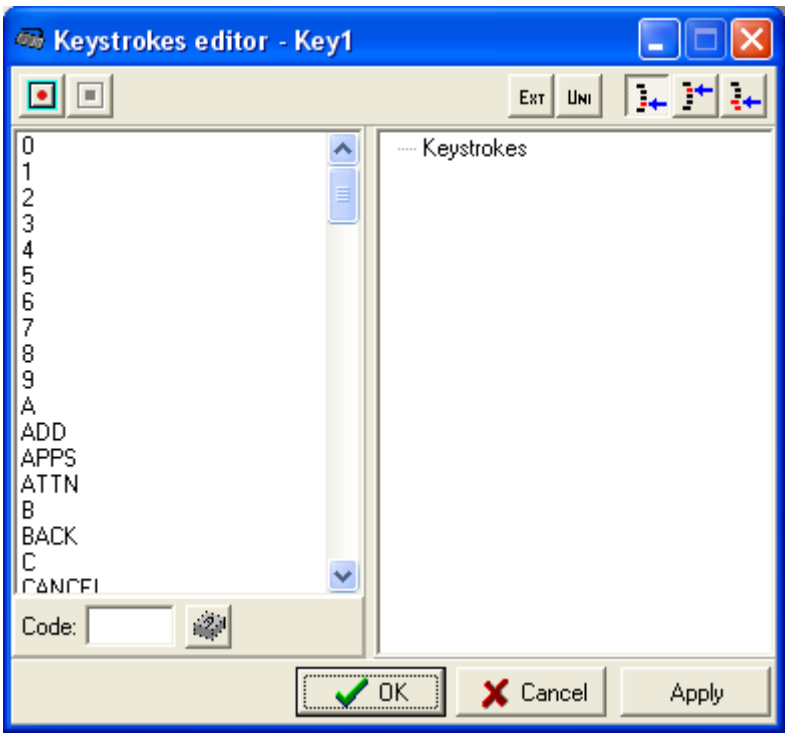
Since we added the Caption "Hi!", let's make the key type that when we click on it.

1. Click on the key to select it, find the key's Keystroke's property and click on the property value to the right to make the ellipsis button appear. Click again on the ellipsis button to show the Keystrokes Editor.



© 2002 MountFocus Information Systems

You should now have a new window with two frames. The left side shows a list of keystrokes available and the right side shows the keystrokes in tree view which this key will perform. That side should be empty for now, since we haven't added any keystrokes yet.

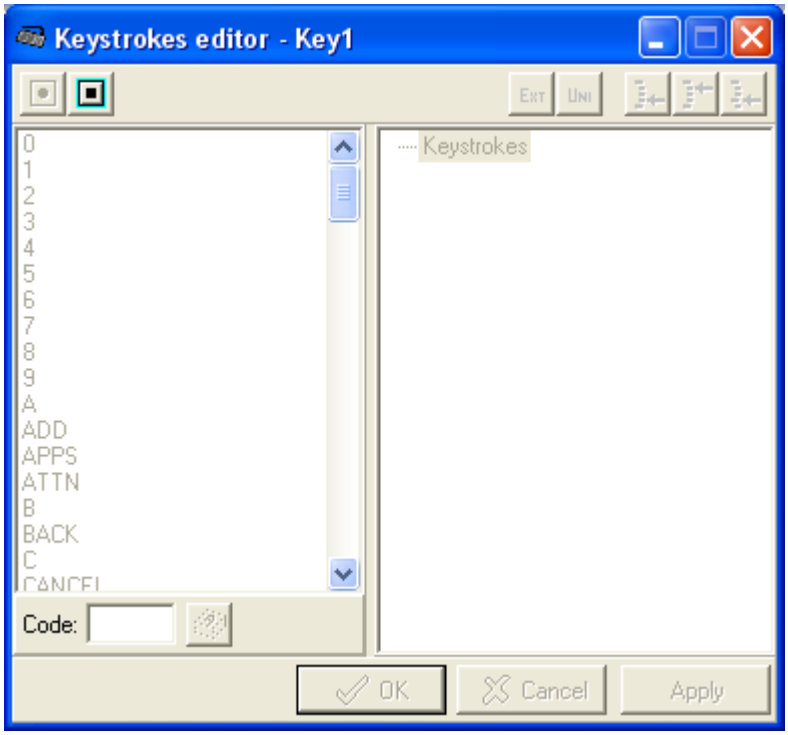


If this key was going to be a key with a single character, the easiest method is to drag the keystroke you want to the word "Keystroke" in the right pane of the frame and you would be done. The key you just created has a phrase, so we're going to use the Keystrokes recorded to make our job easier.

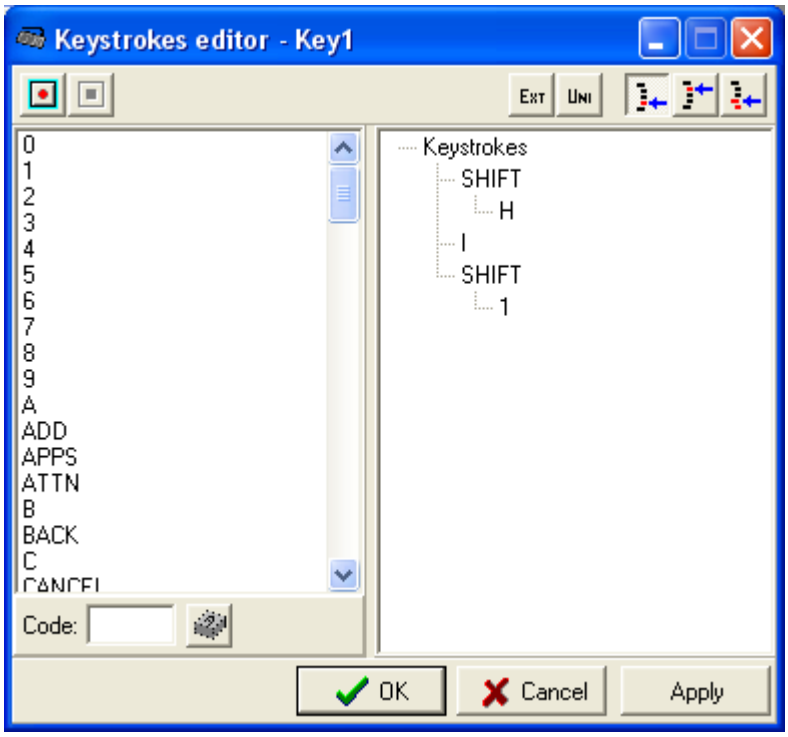
At the top left of the Keystrokes Editor window there are 3 icons. The first plays keystrokes, the second records keystrokes and the third stops recording keystrokes. If you pause your mouse over each icon, you will reveal a Help tip showing the function of each.

© 2002 MountFocus Information Systems

2. Click the Record Keystrokes icon and you will notice that everything in the window becomes disabled except the Stop Recording icon. You can now type as you would on your keyboard and each keystroke will be recorded as you make it and appear on the right pane of the window.



3. Type `Shft + H i + Shift + !` and then click on the Stop Recording icon.



© 2002 MountFocus Information Systems

4. Now you can click on Apply or OK to save the keystrokes you've chosen for your Key.

   Congratulations! You've completed your first key!

3.3.1.6 Adding additional objects

Now that you've learned to place a key on the keyboard, placing different kinds of objects should be quite easy.

The basics are the same.

1. Click on the tool you want to use.
2. Click again on the keyboard surface.
3. Change the properties of the object to your liking.

If you would like to try placing several objects on the keyboard and changing their properties to see the results, this would be a good time to do that.

A more detailed tutorial included in the manual will go into more depth and walk you through the use of each object and property.

Let's test your new keyboard!

3.3.1.7 Testing your Keyboard

The last part of this tutorial will be a test of the simple keyboard you just made.

- First, find the Run button on the toolbar and click on it (or choose File| Run from the drop-down menu).

Once you have done this, your keyboard should appear in the lower right corner of your screen.
Now we'll need to open a program that will use the input from the keyboard.

1. Open Notepad or any other text editor or word processing program you have on your computer.
2. Now click on the key that you made that says, "Hi!"
3. Watch as the application you just opened types out the word, "Hi!"

Congratulations! You just built your first working keyboard and completed this tutorial!

© 2002 MountFocus Information Systems

3.4 Nice to Know

3.4.1 Customizing the Keyboard Designer

This chapter explains some of the ways you can customize the MountFocus Keyboard Designer to suit your needs.

3.4.2 Organizing your work area

3.4.2.1 Arranging windows

When you open the MountFocus Keyboard Designer, the Property Editor is positioned on the left side of the window by default. This window can be moved and re-sized as you wish. The next time you open the Keyboard Designer, the program will remember your previous settings.

Opening a new Keyboard will also open a Script editing window, which can be minimized by clicking on the minimize button or close button in the top right corner of that window.

General defaults for the Keyboard Designer can be set by choosing View | Options from the Menu and choosing the feature that you wish to customize from the tabs available.

You can use this to set the defaults for the Caption Editor, the Script Editor and the Keyboard Editor.

3.4.2.2 Setting Defaults

Two types of defaults can be set:

- System default
- Keyboard default

When you create a new keyboard file, the System Default values are automatically copied to the new keyboard.

All new objects created use the current Keyboard Defaults to set their initial property values. To change the default values for a given object type, right click an object (after setting the property values as you see fit) and select either "Set as system default" or "Set as keyboard default". The Keyboard Default values are only stored when you use the standard "binary" keyboard file format (.kbd/.kbp).

© 2002 MountFocus Information Systems

Part

4 Keyboard Designer Manual

4.1 How To ...

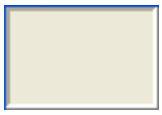
This section contains helpful hints on the most common operations performed in the Keyboard Designer.

4.1.1 Set defaults

The Keyboard Designer keeps two sets of default object values (standard property values for new objects); System Defaults and Keyboard Defaults. When you create a new keyboard file, the System Default values are automatically copied to the new keyboard. All new objects created use the current Keyboard Defaults to set their initial property values. To change the default values for a given object type, right click an object (after setting the property values as you see fit) and select either "Set as system default" or "Set as keyboard default". The Keyboard Default values are only stored when you use the standard "binary" keyboard file format (.kbd/.kbp).

4.1.2 Identify objects

The following tables show the initial appearance of objects as they are created.

Keyboard	Key	Label	Image	Indicator
				

Region	Panel	Page control	Page

4.1.3 Place objects on a keyboard

Placing objects on the keyboard can be accomplished in two ways:

- Choosing the tool from the Tool menu and clicking on the keyboard surface
- Choosing an icon from the Toolbar and clicking on the keyboard surface

Clicking on any tool and moving your mouse over the keyboard will cause the mouse pointer to change to an arrow with a box attached with a letter or graphic on a white background, to let you know that you will be placing an object.

When you select a tool in the Keyboard Designer, it works once (i.e. if you select the key tool and then place a key on a keyboard, the tool reverts to the select tool automatically). To make a tool

© 2002 MountFocus Information Systems

"sticky" (keep the tool selected), click twice on the tool icon on the toolbar.

Clicking once on the keyboard surface with a tool chosen will place an object with the default size values. You can also choose a tool and click and drag to size that object as you place it. The label object has autosize set to true as the default, so this object will automatically resize itself even if you click and drag to make the object larger. Of course you can change the default setting for the label to a different size, font, and set autosize to False if you choose.

Note: the "Create new page" tool is only active if you have already placed a page control on the keyboard surface.

The following table illustrates the relationships between various objects. The named objects in the top row of the table indicate the parent object, and the objects listed below each are the child objects which they can contain.

Keyboard	Key	Indicator	Panel	Page control	Page
Key	Label	Label	Key	Page	Key
Label	Image	Image	Label		Label
Image	Indicator	Indicator	Image		Image
Indicator			Indicator		Indicator
Region			Region		Region
Panel			Panel		Panel
Page control			Page control		Page control
Page			Page		Page

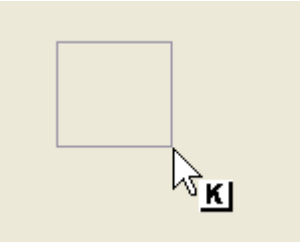
For a full explanation of how to place objects on key/indicator objects, refer to the Caption Editor section.

Note: Label, Image and Region objects are not included in the above table because those objects cannot contain other objects.

4.1.4 Size objects

When an object is placed on the keyboard by clicking once, it is sized according to the keyboard defaults.

An object can also be sized at the time of placement or later. If you wish to make an object larger or smaller than the default size, you can click and drag to the desired size when placing the object, or you can adjust the size in the property editor after placement.



Objects can also be scaled.

© 2002 MountFocus Information Systems

Note: Pages and page control objects are treated slightly differently than other objects because pages cannot be sized, only page control objects. Move objects contains instructions on how to work with page control objects.

4.1.5 Move objects

Objects can be moved around on the keyboard by dragging or changing the position using the property editor.

Note: if more than one object is selected, only the properties they have in common will be displayed.

Select objects by clicking once to select one object, clicking and dragging a rectangle to include multiple objects, or by Shift clicking to select multiple objects.

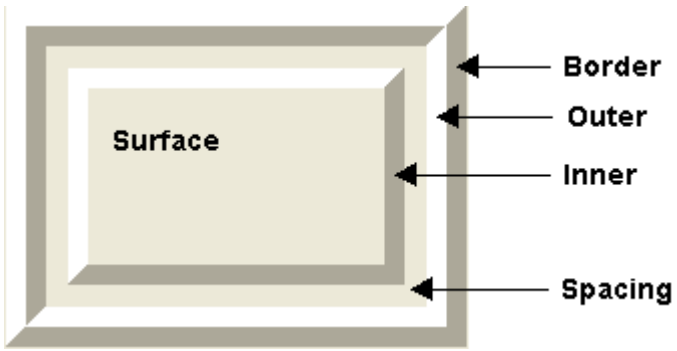
Moving or sizing a page on a page control object is only possible by selecting the Page control object and making the desired changes. Select pages on the page control object by right clicking and choosing Select page from the menu and then select the page you wish to edit, or select None to edit the page control object.

Alternatively to edit the page control object, you can select the page object and press Esc until the page control is showing and then make any changes you wish. To reveal the page objects again, right click on the page control object and choose Select page from the menu and then select the page you wish to edit.

4.1.6 Create bevel effects

Most controls (windowed and non-windowed) can have one or more bevels. A bevel is a 3D frame that is drawn around the control. By changing the different properties of the bevel you can change the visual 3D effect of the control. The TmfKBBevel class has no events.

The two main parts of a bevel are the Inner and Outer frames. They both provide the same functionality and as their names suggest, one is drawn inside of the other. In addition there is a third frame, Border, which is drawn outside of everything else. Each of the frames can be set independently of the others - both in style and width (thickness), but they all use the same colors.



The image above illustrates the different parts of the bevel property. The top and bottom colors can be changed and several different variations of bevel effects can be combined to create very different effects.

Note: the space area takes on the color of the surface and cannot be changed.

© 2002 MountFocus Information Systems

4.2 Objects Explained

This section contains an explanation of each type of object.

4.2.1 Keyboard

Every keyboard file has one, and only one, keyboard object. The main tasks of the keyboard object are to provide a surface for other objects to be placed on, to define the size of the Runtime Keyboard window and to be a placeholder for global settings. Note that you cannot move the keyboard object. It is always positioned in the upper left corner of the Runtime Keyboard window. You can, of course, move the Runtime Keyboard window.

4.2.2 Key

The basic purpose of the key object is, of course, to send keystrokes to Windows. However, the key object can also perform a number of other tasks such as changing pages on a page control, allowing movement of the Runtime Keyboard or shutting down the Runtime Keyboard. In addition you can write scripts that handle the events of the key.

The following events occur, in the order listed, for the key object when the left mouse button is pressed and released over the key object:

- OnButtonDown
- OnKeyPressed
- OnButtonUp
- OnKeyReleased

4.2.3 Label

Label objects are used to display text. Labels can have multiple Lines of Text and can be displayed using any available Font. It is recommended that you use standard fonts (fonts that come with Windows) since only the name of the font is stored in the keyboard file. If the font you use is not available on a system running your keyboard, Windows will replace it with another font and your keyboard may look very different from what you intended.

Label text can be entered directly into the property editor using the Text property, which will create a single line of text. Unicode can also be entered directly into the property editor using the following method:

`&<code>;`

<code> can equal a number or $ plus a HEX value. (eg: `&3053;` for the Hiragana letter KO or `&$221A;` for the square root symbol)

The font chosen should support Unicode for correct display of Unicode characters.

Multiple lines of text can be entered using the Strings editor which is accessed by clicking on the ellipsis button next to the Lines property.

Important: When a label is placed on the caption of a key or an indicator, its events are not triggered since all mouse related events are sent to the owner (the key or indicator it is placed on).

© 2002 MountFocus Information Systems

4.2.4 Image

Image objects are used to hold a bitmap file to allow you to display graphics on your keyboards. You can load image files in bitmap (.bmp), icon (.ico) or Windows Metafile (.wmf or .emf) formats, but bitmaps are recommended as they can be manipulated more easily. Icon files are transparent (regardless of the setting of the Transparent property) and they do not allow stretching (sizing of the graphics object). Metafiles can be sized, but cannot be transparent. Bitmaps can be sized and can be transparent or opaque.

All image files used are stored in the keyboard file. You do not need to distribute the image file along with your keyboard.

Important: When an image is placed on the caption of a key or indicator, its events are not triggered. All mouse related events are sent to the owner (the key or indicator it is placed on). To handle the OnStateChanging and OnStateChanged events, create an indicator state object to handle the events and set the indicator object's MonitorIndicator property to the name of this indicator state object.

4.2.5 Indicator

An indicator object holds two separate captions and switches between them based on its State property. When the State property is set to True the indicator object displays its OnCaption, and when the State property is False it displays its OffCaption. You can set the State property from script or you can use the MonitorIndicator property to link the indicator object to an indicator state object (TmfKBIndState).

Important: When an indicator is placed on the caption of a key or another indicator, its events are not triggered. All mouse related events are sent to the owner (the key or indicator it is placed on). To handle the OnStateChanging and OnStateChanged events, create an indicator state object to handle the events and set the indicator object's MonitorIndicator property to the name of this indicator state object.

4.2.6 Region

A region object defines an invisible area on the keyboard, so you can use regions to create a keyboard that is non-rectangular. Note that you can place a region object on a page. If you do, the region will only be active (create an invisible area) when the page is active.

The region object has no events.

4.2.7 Panel

Panel objects are intended for situations where you want to group some objects and optionally surround the group with a 3D border or different background color. Moving a panel object will also move all its child objects. The TmfKBPanel class publishes all properties and events defined in its ancestor classes.

© 2002 MountFocus Information Systems

4.2.8 Page control

Page control objects are a special kind of surface objects that can not have any child controls except page objects (TmfKBPage). All pages contained within a page control object are sized and positioned according to the size and position of the page control object. Only one page may be visible at any given time. The Bevel and Color properties, as well as all the events, defined in the TmfKBSurface class are not available for page control objects. Instead you have access to these via the TmfKBPage class.

At runtime, the page control object keeps track of all page changes that occur and provides the ability to backtrack via the ClosePage function.

When a page change occurs the following events are triggered in the order listed.

- TmfKBPageControl.OnPageChanging
- TmfKBPage.OnDeactivate (for the page object active before the change)
- (Now the page change takes place)
- TmfKBPage.OnActivate (for the page object becoming active)
- TmfKBPageControl.OnPageChanged

4.2.9 Page

A page object represents a single page on a page control object. A page cannot be sized or moved, since its size and position is determined by the page control object. You can navigate through multiple pages and the page control object by using the right click menu when a page is selected.

4.3 The Keyboard Designer Interface

You can start the Keyboard Designer in several ways:

Double-click the Keyboard Designer icon (if you've created a shortcut).

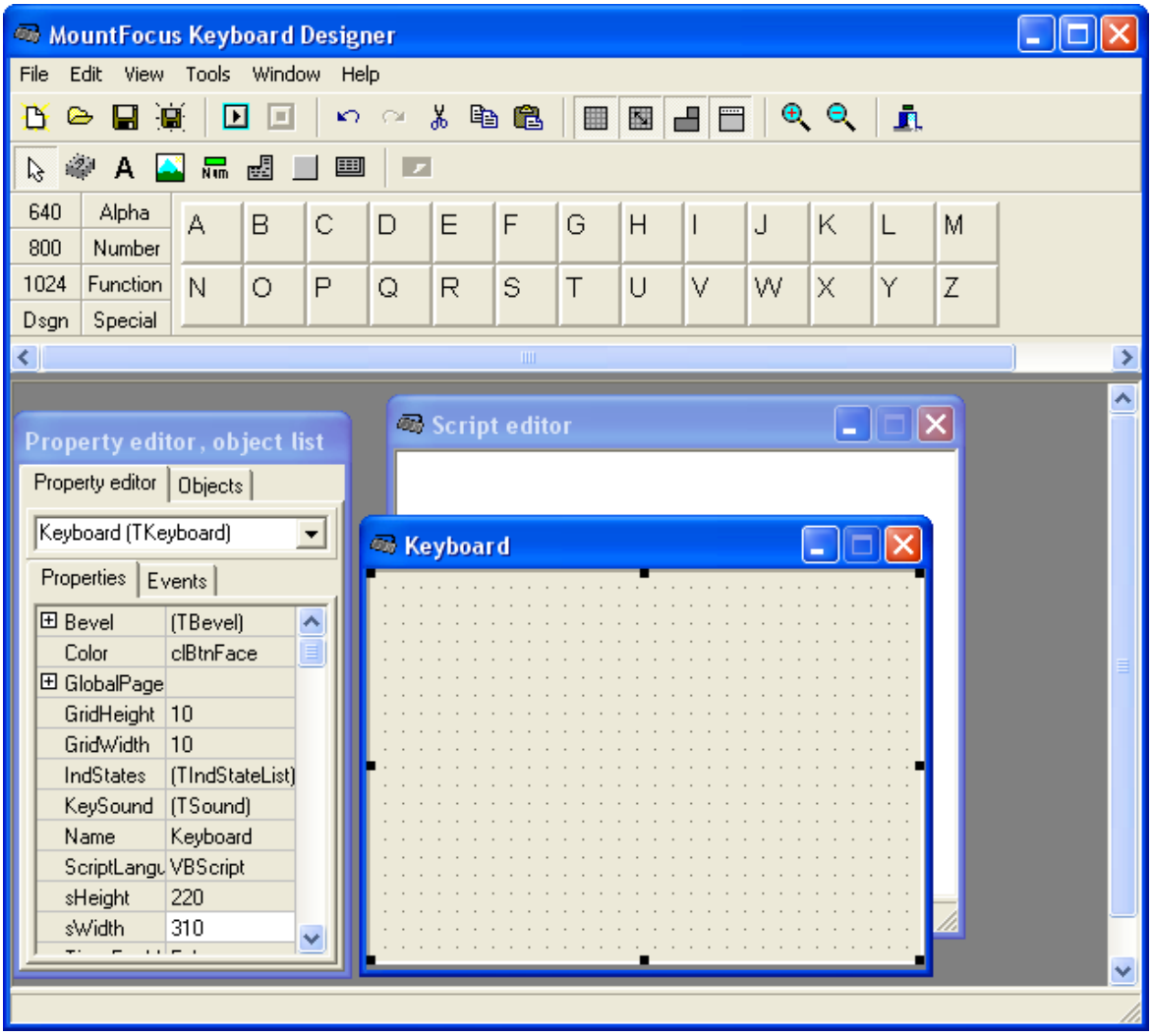
Choose Programs | MountFocus POS | Keyboard Designer from the Windows Start menu.

When the program is started, the Keyboard Designer main window is divided into 4 main sections:

Menu Items
Toolbars
Keypool
Property Editor, object list

All of the above items will be displayed when the Keyboard Designer is opened.

© 2002 MountFocus Information Systems



Each of the sections and their functions are described in detail in separate topics of this help file.

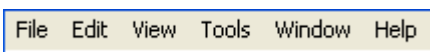
4.3.1 Menu Items

This section contains a description of each menu item available and how to use them, either by choosing them from the menu or the available toolbars.

Many operations have keyboard shortcuts as well as toolbar buttons. When a keyboard shortcut is available, it is always shown next to the command on the drop-down menu.

Mousing over an icon on the toolbar will display a short description of what action the tool will perform.

You can right-click on most objects to display a menu of commands appropriate to the object you are working with.



© 2002 MountFocus Information Systems

4.3.1.1 File Menu

New
Create a new keyboard file
The new file will be named Keyboard until you save the file and rename it.

Open
Open keyboard file
Keyboard files can be saved in binary or text format. Keypool files can also be opened, created and edited just like normal keyboard files. Select the format of the file you wish to open from the "Files of type" drop down box.

Save
Save current keyboard file
Keyboard files can be saved in binary, text or keypool format. Select the format of the file you wish to open from the "Files of type" drop down box.

Save As
Saves the active file with a new name or new type

Close
Closes current keyboard file

Run file
Run the current keyboard file
This is used to test your keyboard using the Runtime Keyboard program. Note that you should close the Runtime using the Close Runtime command and not manually, as this will cause an error.

Close runtime
Close the Runtime Keyboard
Note that you should close the Runtime using the Close Runtime command and not manually, as this will cause an error.

Set Keypool
Select a keypool file to use
This will open a dialog box allowing you to choose a keypool file to be displayed in the default keypool area.

Exit
Quits the MountFocus Keyboard Designer application

4.3.1.2 Edit Menu

Undo
Reverts the last action
The Undo command will undo the last action taken, except the creation of objects. Undo is local to the window, so if you open the caption editor and make changes to the caption, you can undo and redo them as long as that window is open. As soon as you close that window the undo and redo commands for the caption become unavailable.

Redo
Cancel the last undo action
This works in the same way as the Undo command in that it is local to the window as explained above.

© 2002 MountFocus Information Systems

Cut
Cuts the selection and copies it to the Clipboard

Copy
Copies the selection to the Clipboard

Paste
Inserts Clipboard contents
Note that when an object is copied and pasted, it is pasted on top of the original object.

Select All
Selects all objects
Multiple objects can also be selected by shift clicking on the objects or by dragging an area on the keyboard which encloses all objects to be selected. Note that selecting objects on a page on a page control or objects on a panel can also be selected by shift clicking, but to drag an area on the page control or panel you will also need to hold down the Ctrl key while dragging.

Delete
Erases the selection
Objects that are deleted can be reinstated using the Redo command.

4.3.1.3 View Menu

Show grid
Show grid on keyboards
Toggles between making the grid visible or invisible.

Snap to grid
Move and size objects according to the grid
This is useful for making sure objects are lined up correctly or can only increase or decrease in size or position according to the grid settings. Grid setting determine how objects will move when Snap to grid is set. If you set the grid to 5 x 5, objects will move in increments of 5 pixels from the starting point of where the object was first placed. To move objects in single pixel increments, use the arrow keys.

Show regions
Show region objects
Toggles between making regions visible or invisible.

Zoom
Changes the aspect of the view
You can also zoom in and out by using the mouse wheel.

Show keypool
Show or hide the keypool panel
Toggles between making the keypool visible or invisible.

Options
Display program options dialog
The options dialog allows you to set defaults for the keyboard, caption and script editors as well as general defaults.

© 2002 MountFocus Information Systems

4.3.1.4 Tools Menu

When choosing a tool to place an object, note that selecting the tool from the menu or clicking once on the tool icon will allow you to place one object on the keyboard or other surface or control. Double clicking on the tool icon will allow you to place multiple objects of that type. To switch the tool off, click on the select tool icon or choose the select tool from the tool menu.

The only exception to this is the icon to create a new page. This will only be active if a page control has been placed on the keyboard and clicking the icon multiple times will create multiple pages on the page control.

Select tool
Use this tool to select objects
Objects can also be selected by Shift clicking on the objects or by dragging an area on the keyboard which encloses all objects to be selected. Note that selecting objects on a page on a page control or objects on a panel can also be selected by shift clicking, but to drag an area on the page control or panel you will also need to hold down the Ctrl key while dragging.

Key tool
Use this tool to create key objects
Keys can be placed directly on the surface of the keyboard, on panels, or on pages contained on page controls.

Label tool
Use this tool to create labels
Labels can be placed directly on the surface of the keyboard, on panels, pages contained on page controls, and on keys and indicators using the caption editor.

Image tool
Use this tool to create image objects
Images can be placed directly on the surface of the keyboard, on panels, pages contained on page controls, and on keys and indicators using the caption editor.

Indicator tool
Use this tool to create indicators
Indicators can be placed directly on the surface of the keyboard, on panels, pages contained on page controls, and on keys using the caption editor.

Region tool
Use this tool to create a new region object
Regions can be placed directly on the surface of the keyboard, on panels and pages contained on page controls.

Panel tool
Create a new panel object
Panels can be placed directly on the surface of the keyboard, on other panels and pages contained on page controls.

Page control tool
Create a new page control object
Page controls can be placed directly on the surface of the keyboard, on panels and on pages contained on page controls.

Create new page
Create a new page on a page control

© 2002 MountFocus Information Systems

4.3.1.5 Window Menu

Arrange
Arranges the minimized windows to sit at the bottom of the main window if the main window is resized.

4.3.1.6 Help Menu

Opens this help file.

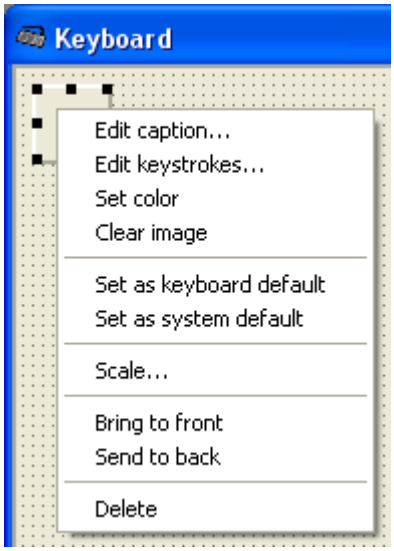
4.3.1.7 Right Click Menus

Right click menus are designed to give easier access to certain properties and functions of objects.

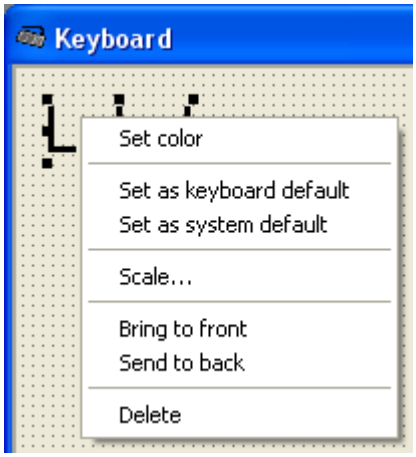
Each object has unique and standard right click menu properties to help you easily change properties or navigate through multiple objects such as in the case of the page control.

The screenshots below show the right click menu for each object:

Keyboard object



Key object



Label object

© 2002 MountFocus Information Systems

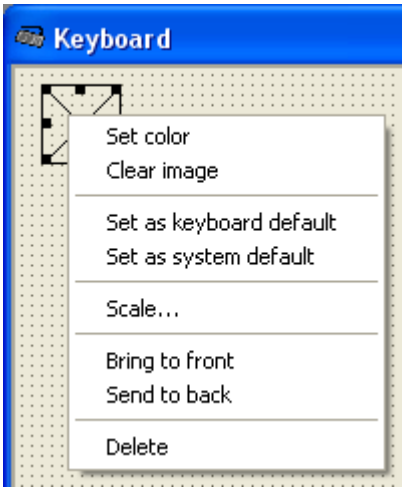
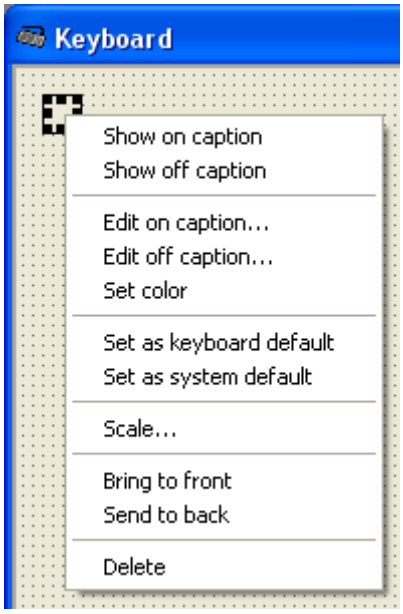


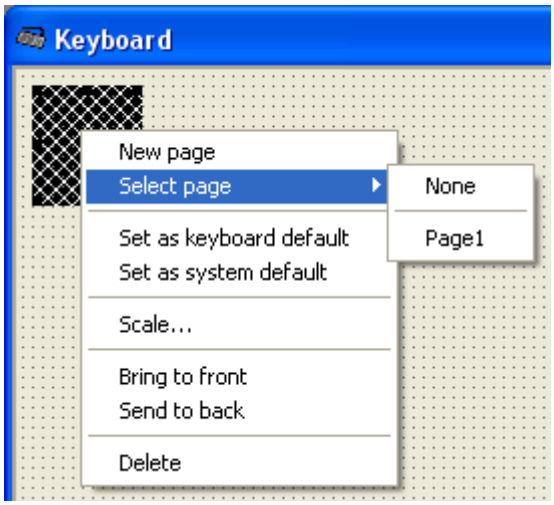
Image object



Indicator object

Region object

Panel object



Page control object

4.3.2 Toolbars

There are two toolbars available in the Keyboard Designer.

The primary toolbar consists of several standard buttons and some specialized buttons:

The secondary toolbar contains the tools to create objects on the keyboard.

© 2002 MountFocus Information Systems

When you select a tool in the Keyboard Designer, it works once (i.e. if you select the key tool and then place a key on a keyboard, the tool reverts to the select tool automatically). To make a tool "sticky" (keep the tool selected), click twice on the tool icon on the toolbar.

All the functions of these toolbars are explained in the Menu Items section.

4.3.3 Keypool

The Keypool is a collection of ready made objects which can be copied to a Keyboard to save time.

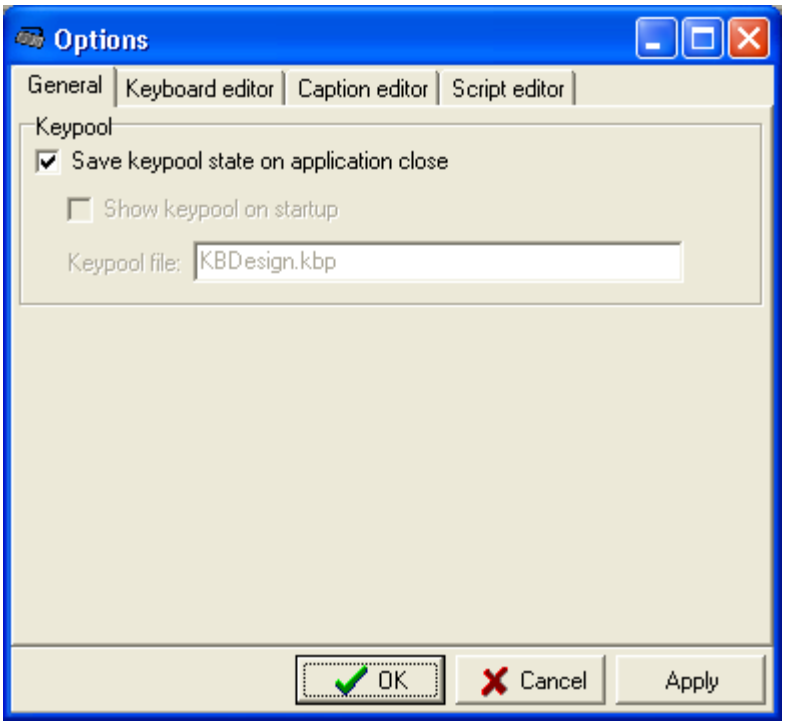
To copy an object simply click on the object in the keypool and again on the keyboard to place it.

Clicking on the Alpha, Number, Function and Special keys allows you to change the available keys for copying. There are two sizes of keys available: those optimized for 640 x 480 resolution and those optimized for 800 x 600 resolution.

Keypool files can also be opened, created and edited just like normal keyboard files.

4.3.4 Options

View options for the Keyboard Designer can be set from the View menu.



© 2002 MountFocus Information Systems

General
Contains the settings for the keypool to be displayed and allows the keypool file to be selected. This is where you can choose a keypool file to be loaded, and set whether or not you want the keypool to be visible on startup.

Keyboard editor
Contains the default settings for displaying the Keyboard. If these are checked, the grid settings will be on.

- Show grid - the grid will be visible on keyboards by default
- Snap to grid - move and size objects according to the grid settings

Caption editor
Contains the default settings for displaying the Caption editor. If these are checked, the grid settings will be on.

- Show grid - the grid will be visible in the caption editor by default
- Snap to grid - move and size objects in the caption editor according to the grid settings
- Zoom - Changes the default zoom aspect of the view

Script editor
Contains the settings for displaying the Script editor.

- Fonts
  - Normal font
  - Keyword font
  - Comment font
  - Number font
  - String font

- Script Language
  - Constants
  - Strings
  - Editor scripts

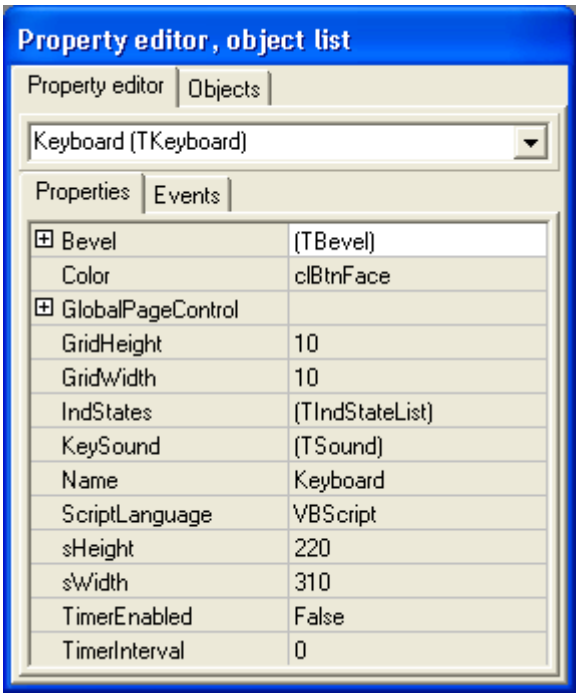
4.3.5 Editors

This section provides an explanation of all the editors the Keyboard Designer uses to create objects on the keyboard.

© 2002 MountFocus Information Systems

4.3.5.1 Property Editor

The property editor is the most frequently used of all the editors and changes according to which object is selected.



Many objects have common properties and a detailed explanation of every property can be found in SurfaceClasses and ControlClasses.

Some properties have simple values -- such as True or False. Others have associated property editors to set more complex values. When you click on such a property value, you'll see an ellipsis. Still others have a default value which can be changed by simply typing a new value in the box.

Some properties can be expanded by clicking on the 'plus' sign to the left of the property. These properties will only expand if the object which uses this property has been selected.

The Property Editor Window is divided into two tabbed sections. The first is the Property Editor. This also has two tabbed sections

Properties
Shows a detailed list of properties for the object or objects selected. If more than one object of differing types is selected, the properties they have in common are shown.

Events
Shows a list of events available which can be triggered by a script.

The second tabbed section is:

Objects
Shows a tree view of the keyboard and all objects on the keyboard. Clicking on the name of the object will select that object on the keyboard.

© 2002 MountFocus Information Systems

4.3.5.1.1 Events

Events are available for certain objects, and the events change according to the object selected. If multiple objects are selected, the events they have in common are displayed.

Events must be scripted.

Double-clicking on the edit box of an event will cause the event to be pasted into the Script Editor for that object. Scripts can then be edited from the Script Editor to suit your needs.

See Writing Scripts.

4.3.5.1.2 Objects

The objects on a keyboard are listed on the Objects tab in tree view.

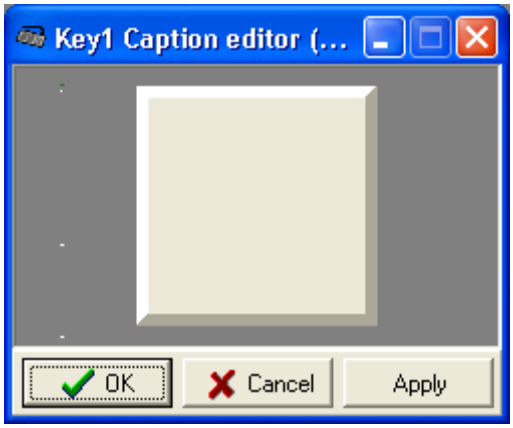
Click to select an object.

See also TmfKBSurface and TmfKBControl.

4.3.5.2 Caption Editors

The Caption Editors are used to add captions to keys and indicators. A caption is simply an object placed on the surface of a key or indicator. For a full explanation of objects and child objects, please see "How to ... Place objects on a keyboard."

There are two ways to open the Caption editor. You can select the key or indicator, and click on the Caption property in the property editor and click on the field to the right. (The indicator has two captions, on and off which are listed separately in the property editor) You should now see an ellipsis button. Click on the ellipsis button until the Caption editor opens. You can also right click on the key or indicator and choose Edit caption from the menu (again, the indicator has two choices, Edit on caption or Edit off caption).



There are three types of captions which can be added to keys and indicators.

- Labels
- Images
- Indicators

Click on one of the three tools and click on the key presented in the Caption Editor to add the object to the key. The properties of the object will appear in the property editor and any changes can be made. Any combination of more than one object can be placed on keys and indicators.

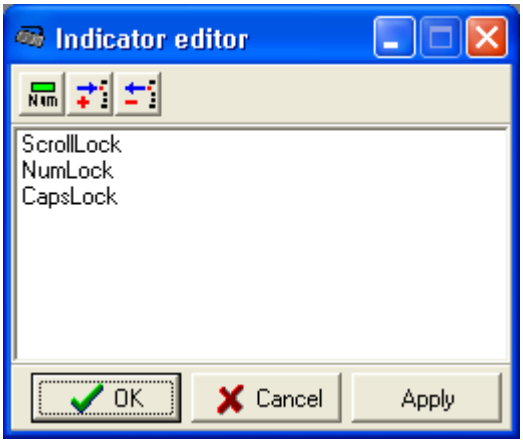
© 2002 MountFocus Information Systems

Important: When a caption is place on a key or indicator, its events are not triggered. All mouse related events are sent to the owner (the key or indicator it is placed on). To handle the OnStateChanging and OnStateChanged events, create an indicator state object to handle the events and set the indicator object's MonitorIndicator property to the name of this indicator state object.

4.3.5.3 Indicator (State) Editor

The Indicator (State) Editor can be opened from the Properties list of the keyboard. Click on the IndStates property in the property editor and click on the field to the right. You should now see an ellipsis button. Click on the ellipsis button until the Indicator (States) editor opens.

The Indicator (State) editor is used to create standard and custom indicators. An indicator object links to the Indicator state object.



There are three pre-defined indicators, which monitor:

1. Scroll Lock
2. Num Lock
3. Caps Lock

There are three icons at the top of the window

Icon	Action
	Create new standard indicator
	Create new custom indicator
	Delete selected indicator

Properties for the Indicators will be displayed in the Property Editor.

- IndNo

Sets a numeric ID for this indicator state. Pre-defined values are:

Scroll Lock -3
Num Lock -2
Caps Lock -1

© 2002 MountFocus Information Systems

Indicators with numeric ID's from 0 to 250 are user defined indicator states and can be switched on and off from scripts or from COM. The numeric value is used by the Runtime Keyboard as an identifier, not the name of the indicator.

- Name

Sets the name of the indicator by typing directly into the edit box.

- State

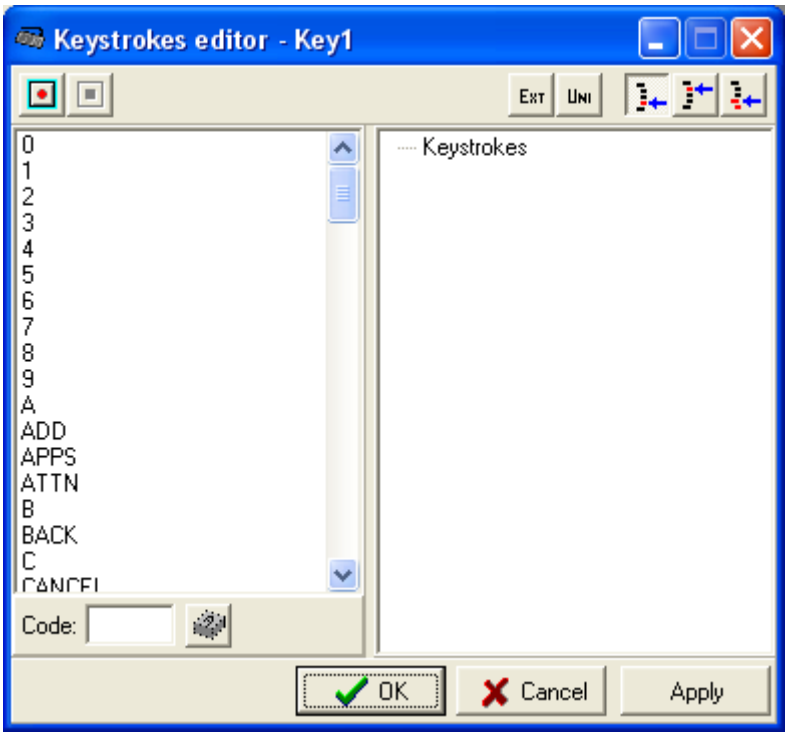
Sets the state of the indicator to True or False.

Clicking on the Create new custom indicator icon will add a new indicator to the list, and clicking on the name will display the properties in the property editor which can then be adjusted according to your needs.

Changes will only take effect after clicking on Apply or OK. Clicking on cancel will cancel all changes made.

4.3.5.4 Keystrokes Editor

The Keystrokes Editor can be accessed from the properties list of a selected key by clicking on the property Keystrokes and then on the ellipsis button to the right of the property.



At the top left of the Keystrokes Editor window are two buttons:

Record keystrokes

Start recording keystrokes from the physical keyboard. This can be useful for recording multiple keystrokes to save time. Also, if you do not know the virtual key name to choose it from the list, you can record it and it will automatically enter the correct virtual key name. Note that the Tab key will not record and the Alt key needs to be held until it appears in the keystrokes tree.

© 2002 MountFocus Information Systems

Stop recording
Stop recording keystrokes.

At the top right of the Keystrokes Editor are five buttons:

Set extended flag

Clicking on this will set the extended flag for a keystroke. Note that Unicode characters are not allowed to have the extended flag set.

Mark as Unicode

Clicking on this will mark a keystroke as Unicode.

Insert as last child
Dropped keystrokes are inserted as the last child of the node you drop them on

Insert as first child
Dropped keystrokes are inserted as the first child of the node you drop them on

Insert as next sibling
Dropped keystrokes are inserted as the next sibling of the node you drop them on

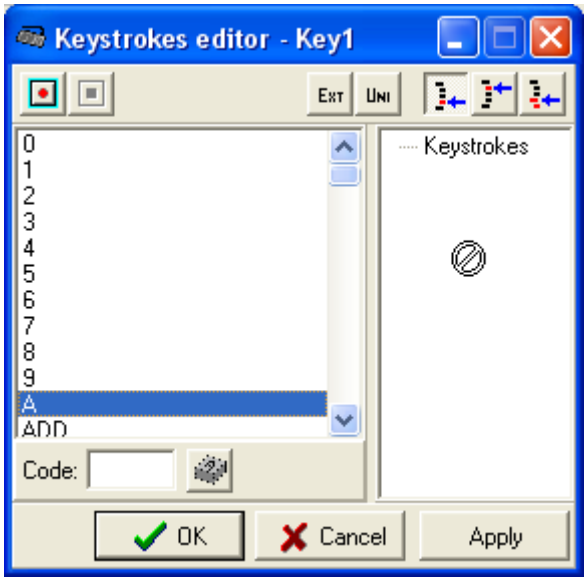
Keystrokes will be dropped according to the method chosen by clicking on one of the buttons described above. The only exception is when recording keystrokes. In that case, the order of keystrokes is not affected by which button is chosen. However, if you re-arrange recorded keystrokes, they will be re-arranged according to the button chosen.

The Keystrokes Editor is divided into two sections:

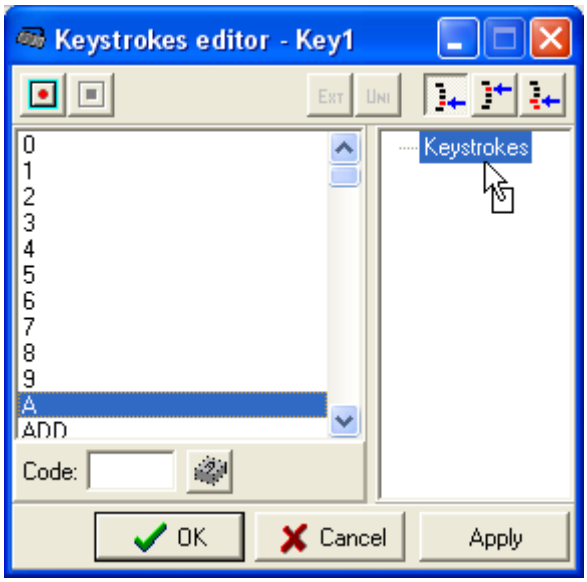
- Keystrokes list
- Keystrokes preview

The keystrokes list shows the standard Windows keystrokes available. These can be dragged to the keystrokes preview area to the right. Note that items must be dragged to the word "Keystrokes" in the preview pane or to any other existing item on the preview list in order to be added to the preview area correctly.

© 2002 MountFocus Information Systems



The picture above shows what the cursor looks like when it is not in the correct position for dropping a keystroke into the preview area.



The picture above shows what the cursor looks like when it is in the correct position for dropping a keystroke into the preview area.

There is also a small field labeled "Code:" This can be used in two ways.

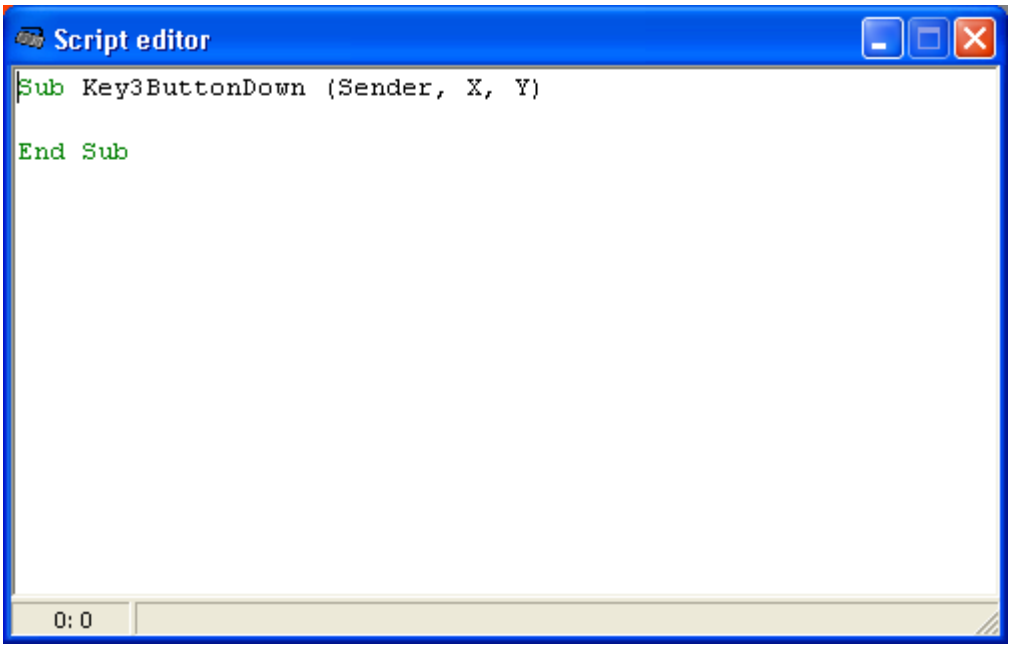
- Clicking on the field and then on the key button to the right of the field allows you to find out the code for any key on the physical keyboard. You can then drag the code using the right click button to the Keystrokes preview area to be added to the list of keystrokes.

- Codes for keys can be typed directly into the field and right clicked and dragged to the Keystrokes preview area as above. This is useful if you need to enter the code for keys which are not on the Keystrokes list, but for which you already know the code (such as Unicode characters). The hexadecimal code preceded by a $ sign are accepted as well as decimals.

© 2002 MountFocus Information Systems

4.3.5.5 Script Editor

The Script Editor is used to build scripts to be used in connection with your keyboard.

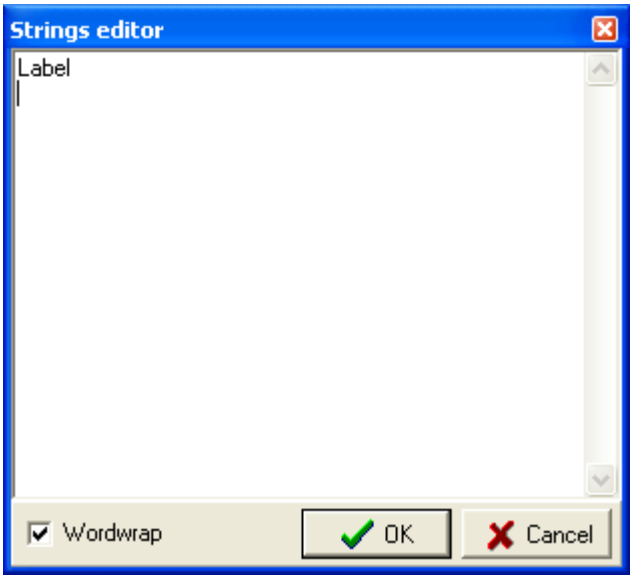


It will open when the Keyboard Designer is opened. Clicking on the close button will minimize the Script editor.

See Writing Scripts.

4.3.5.6 Strings Editor

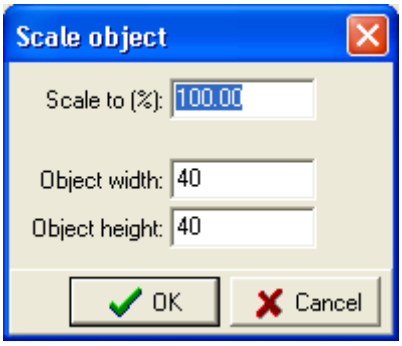
The strings editor is used to enter multiple lines of Label text.



© 2002 MountFocus Information Systems

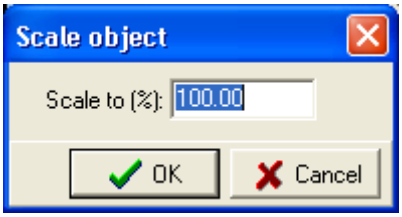
4.3.6 Scale

All objects can be scaled by selecting the object, right clicking and choosing scale from the menu.



Scale can be set by selecting a percentage or by changing the width and/or height of the object.

When multiple objects are selected, scale can only be set by percentage.

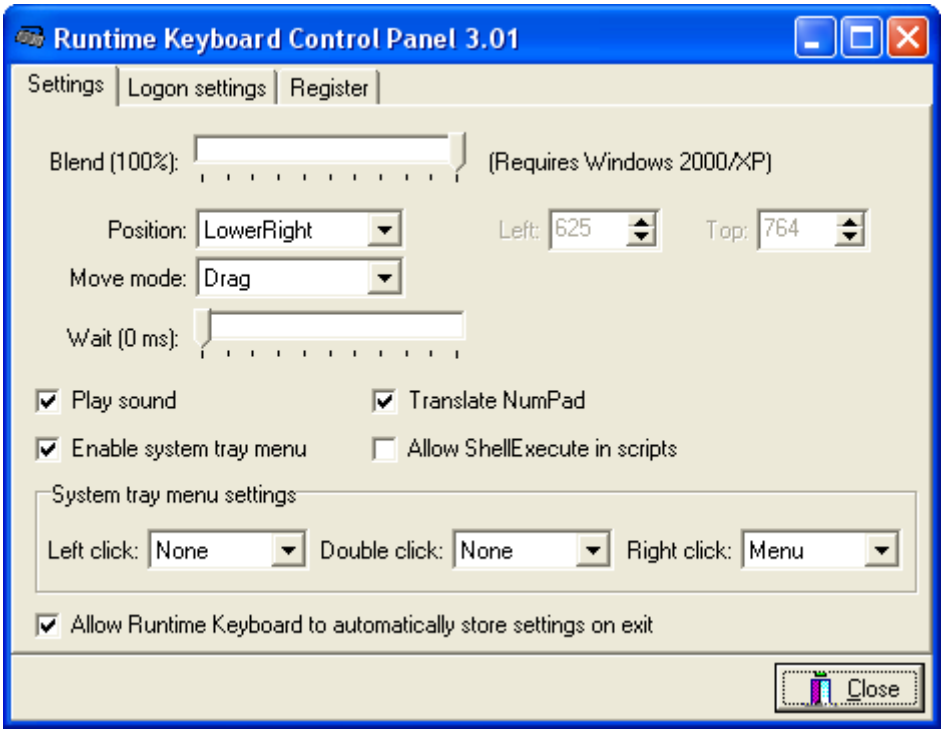


© 2002 MountFocus Information Systems

Part V

5 Runtime Keyboard Control Panel

The Runtime Keyboard Control Panel allows easy access to various settings for customization of the Runtime Keyboard, access to the logon replacement feature as well as the ability to register your copy of the Runtime Keyboard.



Settings tab

Blend

This property allows you to set the blend percentage (requires Windows 2000/XP) to allow the keyboard to be partially transparent such as in this example: (Note that setting the blend to 0% will make the keyboard invisible.)

Position

The following positions can be set:

- Anywhere - allows you to move the Runtime Keyboard once, either by dragging or clicking

© 2002 MountFocus Information Systems

- UpperLeft
- UpperRight
- Center
- LowerLeft
- LowerRight
- Moveable - allows you to move the Runtime Keyboard anytime by dragging only

Using the Left and Top properties you can also set a specific position on the screen (measurement in pixels)

Wait

This property allows you to set a period of time in milliseconds (from 0 to 1000) to wait before sending keystrokes.

Play Sound

If this property is checked, keyboards with sound files associated with objects will be audible. Unchecking this property will not allow sound files to be played.

Translate NumPad

If this property is checked, 0 - 9 and the decimal key of the NumPad are treated as numeric keys when NumLock is on and treated as arrow and positioning keys when NumLock is off. Unchecking this property forces the NumPad keys to remain numeric regardless of whether or not NumLock is on or off.

Enable system Tray menu

If this property is checked the system tray menu is available to the keyboard user. Unckecking this property hides the system tray menu from the keyboard user.

Allow ShellExecute in Scripts

If this property is checked, the command ShellExecute allows scripts to open a program, file, web page or any other item that can be opened from Windows.

System tray menu settings

These properties allow you to assign actions to 3 types of mouse clicks to use with the system tray. The 3 types of mouse click actions are:

- Left click
- Double click
- Right click

The options available for each action are:

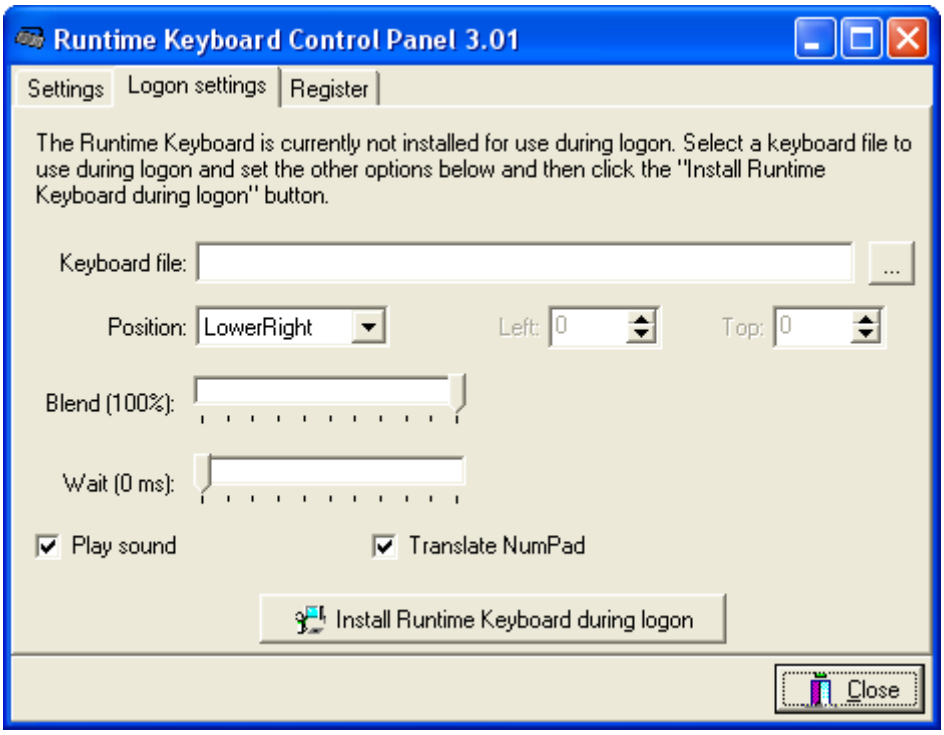
- None
- Menu
- Show
- Anywhere
- Moveable
- Close

Allow Runtime Keyboard to automatically store settings on exit

© 2002 MountFocus Information Systems

If this property is checked, the keyboard settings will be saved on exit. These settings include remembering the last position of the keyboard, remembering which keypool file should be used, whether or not the keyboard is moveable, etc. If this property is not checked, the keyboard will revert to the defaults stored with the keyboard file information each time it is opened and will not remember changes made while the keyboard is open.

Logon settings tab



Note: Changes to the Logon settings and choosing the function Install Runtime Keyboard during logon must be made while logged on as a user with administrator rights. If you are not logged on with administrator rights, these changes will not take effect.

Keyboard file

This field allows you to choose which keyboard file to run at logon.

Position

The following positions can be set for the keyboard used at logon:

- Anywhere - allows you to move the Runtime Keyboard once, either by dragging or clicking
- UpperLeft
- UpperRight
- Center
- LowerLeft
- LowerRight
- Moveable - allows you to move the Runtime Keyboard anytime by dragging only

Using the Left and Top properties you can also set a specific position on the screen (measurement in pixels)

© 2002 MountFocus Information Systems

Blend

This property allows you to set the blend percentage (requires Windows 2000/XP) to allow the keyboard to be partially transparent. (Note that setting the blend to 0% will make the keyboard invisible.)

Wait

This property allows you to set a period of time in milliseconds (from 0 to 1000) to wait before sending keystrokes.

Play Sound

If this property is checked, keyboards with sound files associated with objects will be audible. Unchecking this property will not allow sound files to be played.

Translate NumPad

If this property is checked, 0 - 9 and the decimal key of the NumPad are treated as numeric keys when NumLock is on and treated as arrow and positioning keys when NumLock is off. Unchecking this property forces the NumPad keys to remain numeric regardless of whether or not NumLock is on or off.

Install Runtime Keyboard during logon

Clicking on this button will install the chosen keyboard as the logon replacement.

Register tab

This screen shows the registration status of the software, as well as the about information and how to contact MountFocus Information Systems.

© 2002 MountFocus Information Systems

Part

6 Runtime Keyboard reference

The Runtime Keyboard can execute either as a stand alone program, or as an OLE (COM) automation server. When run as a stand alone program, the Runtime Keyboard is not much different from other Windows programs that you use. You start the program from the start menu or command line, and you are responsible for shutting it down when you no longer need it. An application can use the OLE interface to connect to the Runtime Keyboard and control when the Runtime Keyboard should be available to the user. In this case the startup and shutdown of the Runtime Keyboard is done automatically by the application.

Most of the features of the Runtime Keyboard are available via OLE (COM) and command line parameters. Some are also available through the system tray menu.

6.1 Runtime Keyboard features

The Runtime Keyboard features can be accessed by using command line parameters, OLE (COM) or by writing scripts in your keyboard files. Some features are not available with all of these methods. For example, command line parameters are only read when the Runtime Keyboard starts, so you cannot use command line parameters to perform page changes.

Loading a keyboard file
When you run the Runtime Keyboard as a stand alone application, you can load a keyboard file by specifying the full path and filename as a command line parameter. If the path or filename contains spaces, enclose the filename in double quotes ("Filename"). Alternatively you can specify the "-o" parameter to display the "Open keyboard" dialog box.

The COM interface allows several ways of loading a keyboard file. The first is the LoadKeyboard function which has been available in all versions of the Runtime Keyboard. In version 3.0 three new properties have been added to provide alternate ways of loading a keyboard. The first is the KeyboardFile property which operates exactly like the LoadKeyboard function (actually, using LoadKeyboard with version 3.0 simply sets the KeyboardFile property). Secondly you can use the KeyboardAsBinary and KeyboardAsText properties to read and write the keyboard definition directly. This allows you to store the definition of the keyboard file in a database or within your own files.

You can get the text definition of a keyboard file by saving the keyboard as a keyboard text file (.kbt) in the Keyboard Designer. To get the binary definition of a keyboard file, save the file (in any format) from the Keyboard Designer and then follow these steps:

- Connect to the Runtime Keyboard via OLE
- Use the LoadKeyboard function or KeyboardFile property to load the keyboard file into the Runtime Keyboard
- Read the KeyboardAsBinary property to obtain the keyboard file definition in binary format

These OLE features are not available from script. The reason is that if a script loaded a keyboard file, the script itself would be replaced and the script engine would fail.

Positioning and moving the Runtime Keyboard
You can set the position of the Runtime Keyboard by using the "-p<x>" command line parameter. The <x> is a numeric value from 0 to 6 that will be stored in the Position COM property. When Position is set to Anywhere (0) or Moveable (6), you can also specify the "-l<x>" and/or "-t<x>" parameters to set the Left and Top COM properties respectively - <x> representing the position in pixels from the upper left corner of the desktop. Of course, the command line parameters will only set the initial position of the Runtime Keyboard, while the OLE properties can be used at any time to move the Runtime Keyboard.

© 2002 MountFocus Information Systems

You can also specify the MoveMode property by using the command line parameters "-md" for "Drag" and "-mc" for "Click". Finally, the Height and Width properties can be read via OLE to obtain the Height and Width of the Runtime Keyboard. Since these properties are read only, there are no command line parameters for them.

Showing and hiding the Runtime Keyboard
The OLE methods Show and Hide are used to show and hide the entire Runtime Keyboard window. Version 3.0 also introduces a new OLE property, Visible, which you can read to see if the Runtime Keyboard is visible or not, and write to show or hide the Runtime Keyboard (writing this property will call Show or Hide).

The SetToTop method is used in cases where another Window that has the "Always on top" flag set overwrites the Runtime Keyboard. Calling this method will ensure that the Runtime Keyboard is the topmost window.

You can also read the WinHandle property to get the Windows handle for the Runtime Keyboard window. Using this handle you can call Windows API functions to manipulate the Runtime Keyboard window directly.

These features are not available from the command line.

Changing pages
As you know, the Keyboard Designer allows you to use page control objects that can contain one or more page objects. The purpose of page controls and pages is to allow your keyboard to change its visual appearance and functionality on the fly. In earlier versions there were no page control objects, so all pages covered the entire keyboard. Because of this the OLE methods and properties related to the changing of pages didn't have to worry about which page control to operate on (there was only one set of pages). Now you can have many sets of pages (page controls) so we needed to provide you with the ability to manipulate any page control and at the same time maintain backwards compatibility for applications already using the OLE interface.

The solution was the GlobalPageControl property of the keyboard object. All previous version page change features will operate on the page control object specified in the GlobalPageControl property. These features are the ActivePageIndex and ActivePageName properties and the ClosePage method. In addition the new PageCount property will operate on the GlobalPageControl.

To manipulate any other page control object, you can use the PageControlPageCount, PageControlPageIndex and PageControlPageName properties and the PageControlClose method.

These features are not available from the command line.

Setting indicator states
You can also set the state for any indicator state object that you have defined via the OLE method SetIndicator. Note that only user defined indicator state objects can be manipulated. The system defined indicator states for CapsLock, NumLock and ScrollLock are maintained by the system.

This feature is not available from the command line.

Other settings
The following properties can be set from the command line, via OLE or from script. See the command line topic for details on the format for each one, or follow the link to each property for details on using it via OLE or script.

Blend (Set the level of transparency for the Runtime Keyboard Window)
MenuOn (Enable or disable the system tray menu)

© 2002 MountFocus Information Systems

SoundOn (Enable or disable keyboard sound)
TrNumPad (Enable or disable translation of the numeric keypad keys)
Wait (Set delay before sending keystrokes)

6.2 Command line parameters

The following command line parameters are available when launching the Runtime Keyboard.

Parameter	Description
-exec	Allow scripts to perform ShellExecute commands.
-ne	Disallow scripts to perform ShellExecute commands.
-bl<x>	Set the Blend property. <x> is a numeric value from 0 to 255.
-menu	Set the MenuOn property to 1 (on).
-nm	Set the MenuOn property to 0 (off).
-md	Set the MoveMode property to 0 (drag).
-mc	Set the MoveMode property to 1 (click).
-i<x>	Set the ActivePageIndex property. <x> is the numeric page index to use.
-a<x>	Set the ActivePageName property. <x> is the page name to use.
-p<x>	Set the Position property. <x> is a numeric value from 0 to 6.
-sound	Set the SoundOn property to 1 (on).
-ns	Set the SoundOn property to 0 (off).
-stlc<x>	Set the action to perform when the user left clicks the system tray menu.
-stdc<x>	Set the action to perform when the user double clicks the system tray menu.
-strc<x>	Set the action to perform when the user right clicks the system tray menu.
-num	Set the TrNumPad property to 1 (on).
-nt	Set the TrNumPad property to 0 (off).
-w<x>	Set the Wait property. <x> is a numeric value.
-l<x>	Set the Left property. <x> is a numeric value.
-t<x>	Set the Top property. <x> is a numeric value.
-dl<x>	Set the Delay property. <x> is a numeric value.
-kodt	Set the KeepOnDesktop property to 1 (true).
-nkodt	Set the KeepOnDesktop property to 0 (false).
-o	Display the Open Keyboard dialog on startup.
<Filename>	Set the KeyboardFile property to the filename specified.

© 2002 MountFocus Information Systems

The following values are used for the -stlc, -stdc and -strc command line parameters.

Value (<x>)	Description
0	No action
1	Open system tray menu
2	Show the Runtime Keyboard
3	Set Position to kpAnywhere
4	Set Position to kpMoveable
5	Close the Runtime Keyboard

6.3 System tray menu

By default, the user can open the system tray menu by right clicking the keyboard icon in the system tray. How, and if, the system tray menu is available depends on the MenuOn property and the -stlc, -stdc and -strc command line parameters (these can also be set in the Keyboard Control Panel).

The following commands are available in the system tray menu:

Command	Description
Move	Sets the Position property to the value indicated by the sub command chosen.
Select page	Sets the ActivePageName property to the value indicated by the sub command
Show / Hide	Calls the Show or Hide methods.
Close	Closes the Runtime Keyboard

6.4 Using COM

The Runtime Keyboard can act as an **out of process** OLE server. This means that other applications are capable of controlling the Runtime Keyboard through COM (Component Object Model), but the Runtime Keyboard will always execute as a stand alone application in its own process. This has the benefit that the keyboard will be available for all windows that your application opens, but your application can still manage the keyboard from a central location (i.e. you don't need more than one reference to the keyboard - even if you use it in many different windows).

Connecting using OLE
The Runtime Keyboard OLE server is named "Kbd.mfSoftkeys" and exposes and interface called ImfSoftkeys. How to access and OLE object depends on the development tool you are using. In short, Visual Studio users (version 6.0 and above) can use the `#import` directive (give the Runtime Keyboard executable file - Kbd.exe - as the parameter) to create a C++ class representing the Runtime Keyboard OLE server. Delphi users can use the "Import type library" command on the "Project" menu (search for the Kbd Library) to create a non-visual component that can be used to access the Runtime Keyboard. Visual Basic users can create a reference to the Runtime Keyboard

© 2002 MountFocus Information Systems

server by using the `CreateObject` function. Even though different tools use different methods to connect to an OLE server, the available features and the behavior of the Runtime Keyboard will be the same.

When you create a reference and connect to the Runtime Keyboard OLE server, the Runtime Keyboard application starts automatically. Similarly, when you disconnect, or when the reference is destroyed, the Runtime Keyboard application shuts down. Here is a VB example demonstrating the use of the Runtime Keyboard.

```
' Reference variable to the Runtime Keyboard
' The variable is declared as a global variable
Dim MyKbd

' Call this sub on application startup
Sub Initialize()
  ' Create a reference to the Runtime Keyboard and connect
  MyKbd = CreateObject( "Kbd.mfSoftkeys" )

  ' Now we can access all the methods and properties of the Runtime Keyboard
  MyKbd.LoadKeyboard( "C:\Program Files\MountFocus\Keyboard\US800.kbd" )
  MyKbd.Position = 5 ' Lower right corner
End Sub
```

You can now access the MyKbd variable in any other procedure and use any of the OLE features (properties and methods) to control the Runtime Keyboard. Note that the example above does not disconnect from the Runtime Keyboard. When the application shuts down, the MyKbd variable is destroyed and that will cause the Runtime Keyboard to shut down also.

Connecting using ActiveX

Some tools and applications cannot connect to an OLE server (or makes it very difficult to do so). For this reason the Runtime Keyboard ships with an ActiveX component that you can use instead. The ActiveX control is named "mfSoftkeysControl" and is implemented in the Kbd.dll file that was installed with the Runtime Keyboard. The ActiveX control itself contains no functionality, but simply connects to the Runtime Keyboard OLE server and acts as an ActiveX frontend to the OLE server. So, the features of the OLE server and the ActiveX control are the same.

How to use the ActiveX control also differs depending on the tool you are using, but most tools simply let you place the ActiveX control on your form and then allows the use of its properties and methods.

© 2002 MountFocus Information Systems

6.4.1 COM properties

Properties marked **ro** are read only. Properties marked **v3** are new to version 3.0.

ActivePageIndex
ActivePageName
Blend
Height **ro**
KeyboardAsBinary **v3**
KeyboardAsText **v3**
KeyboardFile **v3**
Left
MenuOn
MoveMode
PageControlPageCount **ro v3**
PageControlPageIndex **v3**
PageControlPageName **v3**
PageCount **v3**
Position
SoundOn
Top
TrNumPad
Visible **v3**
Wait
Width **ro**
WinHandle **ro**

6.4.1.1 COMActivePageIndex

Selects the active page of the GlobalPageControl (0 to number of Keyboard Pages -1).

ActivePageIndex: long

Description
Sets or gets the active page of the GlobalPageControl by numeric index. The first page has number 0, the second page number 1 and so on. The Runtime Keyboard keeps track of all page changes, whether done via OLE, script or key object properties. This allows the ClosePage function to backtrack through the pages.

6.4.1.2 COMActivePageName

Selects the active page of the GlobalPageControl (any valid page name).

ActivePageName: bstr

Description
Works the same way as ActivePageIndex as described previously, except that it works with page names instead of their numeric representation. The page names are not case sensitive.

© 2002 MountFocus Information Systems

6.4.1.3 COMBlend

Sets transparency factor (0 to 255).

Blend: long

Description
Blending is only available when the Runtime Keyboard is running on Windows 2000, Windows XP or later. A Blend value of 0 makes the entire Runtime Keyboard transparent, whereas a Blend value of 255 makes it completely opaque. Values between make the Runtime Keyboard more or less transparent, allowing the application behind the keyboard to be visible at the same time as you use the Runtime Keyboard.

6.4.1.4 COMDelay

Sets global delay between keystrokes in milliseconds.

Delay: long

Description
This property allows a delay to be set between each keystroke. It's useful for sending keystrokes on a slow computer that cannot process keystrokes as quickly, or for opening a menu, choosing an item and then continuing input in the new window.

6.4.1.5 COMHeight

Returns the Runtime Keyboard window height in pixels.

Height: long

Description
This property returns the height in pixels of the full size Runtime Keyboard window. Using regions to hide parts of the active Keyboard Page does not affect this property. The height of the keyboard is set in the Keyboard Designer.

This is a read only property.

6.4.1.6 COMKeepOnDesktop

Forces the Runtime Keyboard to remain on the desktop area or allows it to be moved or dragged outside the desktop area.

KeepOnDesktop: long

Description
Read the KeepOnDesktop property to determine if the Runtime Keyboard remains on the desktop area; True (1), or can be moved or dragged outside the desktop area; False (0). The default is True.

© 2002 MountFocus Information Systems

6.4.1.7 COMKeyboardAsBinary

Holds the binary definition of the keyboard file.

KeyboardAsBinary: bstr

Description
Use the KeyboardAsBinary property to read or write the binary definition of a keyboard file. Use this property to store the binary definition of a keyboard file using your own method of storage (for example, you can store the keyboard definition in a database).

This property is unavailable from script.

6.4.1.8 COMKeyboardAsText

Holds the text definition of the keyboard file.

KeyboardAsText: bstr

Description
Use the KeyboardAsText property to read or write the text definition of a keyboard file. Use this property to store the text definition of a keyboard file using your own method of storage (for example, you can store the keyboard definition in a database).

This property is unavailable from script.

6.4.1.9 COMKeyboardFile

Holds the filename of the current keyboard file.

KeyboardFile: bstr

Description
Read the KeyboardFile property to get the name of the current keyboard file. If the current keyboard has been loaded by using the KeyboardAsBinary or KeyboardAsText properties, the KeyboardFile property returns an empty string. Writing to the KeyboardFile property is equivalent to calling the LoadKeyboard method.

This property is unavailable from script.

6.4.1.10 COMLeft

Sets the Runtime Keyboard left position (0 to desktop width - keyboard width).

Left: long

Description
The Left property refers to the left side of the Runtime Keyboard window, measured in pixels relative to the left side of your desktop. Certain programs, like the task bar or the Office bar, may change the size of the desktop, affecting placement of the Runtime Keyboard.

© 2002 MountFocus Information Systems

6.4.1.11 COMMenuOn

Switches the system tray menu on or off (0 for "off" and any other value for "on").

MenuOn: long

Description
In some cases (especially when using OLE) it may not be desirable to allow the user direct access to the popup menu functions. This option then controls whether or not the system tray icon popup menu is operational or not.

6.4.1.12 COMMoveMode

Selects how to move the Runtime Keyboard in Anywhere mode (0 for "Drag" and 1 for "Click" - other values map to "Drag").

MoveMode: long

Description
The Runtime Keyboard allows two different ways for the user to move the Runtime Keyboard window - Drag and Click. In Drag mode the user must click and drag the Runtime Keyboard, whereas in Click mode a large window covering the entire screen is displayed and the user can move the keyboard with a single click. This can be appropriate for touch screen environments where dragging across a screen is undesirable. The MoveMode setting only has effect when the Runtime Keyboard is in "Move Anywhere" mode (see Position).

In script, use the values "mmDrag" and "mmClick" for this property.

6.4.1.13 COMPageControlPageCount

Returns the number of pages for the page control object.

PageControlPageCount[PageControl: bstr]: long

Description
The PageControlPageCount property takes the name of the page control to operate on as a parameter. If the current keyboard contains a page control object with the specified name, PageControlPageCount returns the number of pages in that page control object. If no matching page control object is found, PageControlPageCount returns -1.

This is a read only property.

6.4.1.14 COMPageControlPageIndex

Holds the current page index for the page control object.

PageControlPageIndex[PageControl: bstr]: long

Description
The PageControlPageIndex property takes the name of the page control to operate on as a parameter. If the current keyboard contains a page control object with the specified name, PageControlPageIndex returns the current page index for that page control object. If no matching page control object is found, PageControlPageIndex returns -1.

You can also change the current page for the page control by writing to this property. If no match is

© 2002 MountFocus Information Systems

found for the PageControl parameter, writing to PageControlPageIndex has no effect.

6.4.1.15 COMPageControlPageName

Holds the current page name for the page control object.

PageControlPageName[PageControl: bstr]: bstr

Description
The PageControlPageName property takes the name of the page control to operate on as a parameter. If the current keyboard contains a page control object with the specified name, PageControlPageName returns the current page name for that page control object. If no matching page control object is found, PageControlPageName returns an empty string.

You can also change the current page for the page control by writing to this property. If no match is found for the PageControl parameter, writing to PageControlPageName has no effect.

6.4.1.16 COMPageCount

Returns the number of pages for the GlobalPageControl object.

PageCount: long

Description
The PageCount property returns the number of pages in the GlobalPageControl object.

This is a read only property.

6.4.1.17 COMPosition

Sets the position of the Runtime Keyboard. Accepts the values listed below. Any other value maps to "Anywhere".

Position: long

Description
The Runtime Keyboard can be moved to five predetermined positions on your desktop, each identified by its own number. Additionally, placing the Runtime Keyboard in "Move Anywhere" mode (Position 0) lets the user move the Runtime Keyboard once, either by dragging or clicking (see MoveMode). Once the Runtime Keyboard is moved it remains stationary until Anywhere mode is entered again.

"Movable" mode (Position 6) allows the user to move the Runtime Keyboard, by dragging only, at any time.

© 2002 MountFocus Information Systems

Position	COM value	Script value
Anywhere	0	"kpAnywhere"
Upper left corner of current monitor	1	"kpUpperLeft"
Upper right corner of current monitor	2	"kpUpperRight"
Center of current monitor	3	"kpCenter"
Lower left corner of current monitor	4	"kpLowerLeft"
Lower right corner of current monitor	5	"kpLowerRight"
Moveable	6	"kpMoveable"

6.4.1.18 COMSoundOn

Switches sound on or off (0 for "off" and any other value for "on").

SoundOn: long

Description
A sound will be played every time the user presses a key on the Runtime Keyboard. The sound itself is a part of the keyboard file, and while some users may find the sound useful (or cool), others may find it utterly annoying. The SoundOn property allows you to turn sound playing on or off.

6.4.1.19 COMTop

Sets the Runtime Keyboard top position (0 to desktop height - keyboard height).

Top: long

Description
The Top property refers to the top of the Runtime Keyboard window measured in pixels relative to the top of your desktop. Certain programs, like the task bar or the Office bar, may change the size of the desktop affecting the placement of the Runtime Keyboard.

6.4.1.20 COMTrNumPad

Switches translation of numeric keypad keys on or off when <Num Lock> is off (0 for "off" and any other value for "on").

TrNumPad: long

Description
The ten numeric keys on the numeric keypad, plus the decimal key, work in different ways depending on whether or not <Num Lock> is active. The Runtime Keyboard detects the <Num Lock> state and translates these keys to their correct counterparts if <Num Lock> is off. To avoid translation (sending numbers independent of the <Num Lock> state), turn off the TrNumPad property.

© 2002 MountFocus Information Systems

6.4.1.21 COMVisible

Holds the visual state of the Runtime Keyboard.

Visible: long

Description
Read the Visible property to determine if the Runtime Keyboard window is visible (1) or hidden (0). Writing to this property is equivalent to calling the Show (1) and Hide (0) methods.

6.4.1.22 COMWait

Sets the time in milliseconds to wait before dispatching keystrokes (0 or higher - values over 1000 renders the keyboard more or less useless).

Wait: unsigned long

Description
In certain, exceptional cases, the Runtime keyboard must wait for a short while before sending keystrokes to the target application. If you experience that keystrokes are sent before the target application regains focus, you can set this property to cause the Runtime Keyboard to wait for the specified amount of time. The wait period is set in milliseconds (ms) which is 1/1000 of a second.

6.4.1.23 COMWidth

Returns the Runtime Keyboard window width in pixels.

Width: long

Description
This property returns the width in pixels of the full size Runtime Keyboard window. Using regions to hide parts of the active Keyboard Page does not affect this property. The width of the keyboard is set in the Keyboard Designer.

This is a read only property.

6.4.1.24 COMWinHandle

Returns the standard windows handle value.

WinHandle: long

Description
This property simply returns the Windows assigned window handle for the Runtime Keyboard window. This can be useful if you want to perform Windows API calls using the Runtime Keyboard window as the target. Naturally, due to the nature of its use, anything you use it for can, and will, be held against you in a support case :-).

This is a read only property.

© 2002 MountFocus Information Systems

6.4.2 COM methods

Methods marked **v3** are new to version 3.0.

ClosePage
Hide
LoadKeyboard
PageControlClose **v3**
SetIndicator
SetToTop
Show

6.4.2.1 COMClosePage

Closes the current page in the GlobalPageControl.

ClosePage

Description
Calling this function via OLE causes the GlobalPageControl to backtrack to the previous page. This can be very useful when your application opens a common dialog window that requires its own page. When the window is activated it can select the page it wants, and when deactivated it can call the ClosePage function and thereby reselect whichever page was displayed previously.

6.4.2.2 COMHide

Hides the entire Runtime Keyboard window.

Hide

Description
Hide (you guessed it) simply hides the entire Runtime Keyboard.

6.4.2.3 COMLoadKeyboard

Loads the keyboard file specified in the Filename parameter.

LoadKeyboard(Filename: bstr)

Description
Calling LoadKeyboard is equivalent to writing to the KeyboardFile property. Make sure to specify the entire path and filename (including extension) in the Filename parameter. When running as an OLE server, the Runtime Keyboard has no "current directory".

6.4.2.4 COMPageControlClose

Closes the current page in a page control object.

ClosePage(PageControl: bstr)

Description
Set the PageControl parameter to the name of the page control you want to operate on. Calling this

© 2002 MountFocus Information Systems

function causes the page control to backtrack to the previous page.

6.4.2.5 COMSetIndicator

Sets the indicator specified in the IndicatorNo parameter (0 to 250) to "off" if the State parameter is 0 and "on" if the State parameter is non-zero.

SetIndicator(IndicatorNo, State: long)

Description
Available only through OLE and scripting, this function allows you to set the state of your user defined indicators. See the indicator state object for details on user defined indicators.

6.4.2.6 COMSetToTop

Forces the Runtime Keyboard window to reset itself to "stay on top", placing it on top of newer windows using the same function.

SetToTop

Description
The Runtime Keyboard stays on top of all other applications. However, if your application also stays on top, you may experience that the Runtime Keyboard ends up behind your application. A call to this function makes sure the Runtime Keyboard appears on top again.

6.4.2.7 COMShow

Redisplays the Runtime Keyboard window if it was previously hidden.

Show

Description
A call to this OLE function redisplays the Runtime Keyboard window (if it was hidden - that is).

6.5 Writing scripts

New in version 3.0 is the ability to write scripts that are executed when certain events occur in the Runtime Keyboard. To define an event handler (script procedure that is executed when an event occur) in the Keyboard Designer, select the object you want to handle an event for and click the Events tab in the property editor. Then double click the event you want to handle and a script procedure prototype is created for you in the Script editor. If you want to name the script procedure manually, type the name you want for the procedure in the event property and press <Enter> instead of double clicking.

You can choose any installed Active Script compatible script language - just make sure that the systems that will run the keyboard files have the same script languages installed. To set the script language to use for a keyboard, enter the name of the language in the ScriptLanguage property. (If you change the script language after you started to create script procedures, you will have to rewrite the existing procedures manually.)

You can find documentation on the VBScript and JScript languages at http://msdn.microsoft.com/scripting. You can also download the latest version of the Active Script

© 2002 MountFocus Information Systems

system from this site.

Accessing keyboard objects
When the Runtime Keyboard starts, the script engine is activated and one object called "Kbd" is added to the script engine. The Kbd object is a reference to the Runtime Keyboard main window and it exposes all the OLE properties and methods of the Runtime Keyboard OLE server - except LoadKeyboard, KeyboardFile, KeyboardAsBinary and KeyboardAsText.

The Kbd object also exposes a property to access the keyboard object. Note that the name of this property is equivalent to the Name property for the keyboard object (that you set in the Keyboard Designer). So, if you leave the standard name for the keyboard object ("Keyboard") you can access the keyboard object from script by typing `Kbd.Keyboard`

If you change the name of the keyboard object to "MyKeyb" you would have to type `Kbd.MyKeyb` to access the keyboard object.

This form of access is called dot notation. You can expand on the statements above in the same fashion. For example if your keyboard has a key with a label on it, called Key1 and Label1 respectively, you can access the label by typing `Kbd.Keyboard.Key1.Label1`

You can also access objects by using the GetControl function of the Kbd object.

6.5.1 Kbd Object

The Kbd object exposes all the properties and methods of the OLE server - except LoadKeyboard, KeyboardFile, KeyboardAsBinary and KeyboardAsText. It also adds two extra methods that are only available from script - not via OLE.

For details on properties, methods and events available for other keyboard objects, refer to the class reference section.

6.5.1.1 Kbd Object methods

GetControl
ShellExecute

6.5.1.1.1 Kbd.GetControl

Retrieve a reference to an object by name. See TmfKBKeyboard.GetControl for details.

6.5.1.1.2 Kbd.ShellExecute

Execute a file.

ShellExecute Operation, File, Parameters, Directory: string

Description
Call ShellExecute to open a program, file, web page or any other item that can be opened from Windows. All four parameters are strings. The first parameter, Operation, can have one of the following values:

"open" Open the file specified in the File parameter. The File can be an executable file, a document, a folder to open or even a web page.

© 2002 MountFocus Information Systems

"print"	Print the file specified in the File parameter. The File should be a document file.
"explore"	Explore the folder specified in the File parameter.

The File parameter is the file to operate on (see table above). If the File is an executable file (a program) the Parameters parameter may contain any command line parameters that you want to add. Finally the Directory parameter is the directory where you want the program to start (default directory).

Note that the Runtime Keyboard must have its AllowExecute setting set to True for the ShellExecute method to work. If the AllowExecute setting is set to False, calls to the ShellExecute method will simply be ignored.

© 2002 MountFocus Information Systems

Part VII

7 Classes

7.1 Surface classes

7.1.1 TmfKBSurface

The TmfKBSurface class is the ancestor for all windowed controls used by the virtual keyboard. It introduces and implements behavior common to these controls, but does not publish this behavior. For example, the Bevel property is defined in this class, but is only made available by the descendants that allow the property to be used.

A surface is a control that can have other controls, called child controls, on top of it. Note that even though you can place labels, images and indicators on top of both keys (TmfKBKey) and indicators (TmfKBIndicator), these two are not surfaces but controls (which is why you have to use the caption editor in the Keyboard Designer to place objects on them).

Surface objects are the keyboard (TmfKBKeyboard), panels (TmfKBPanel), page controls (TmfKBPageControl) and pages (TmfKBPage).

7.1.1.1 TmfKBSurfaceProperties

Properties in *italics* are unavailable and can only be used with descendant classes that publish them.

Bevel
Color
pLeft
pTop
sHeight
sWidth

7.1.1.1.1 TmfKBSurface.Bevel

Specifies a 3D border to be drawn around the surface object.

Bevel: TmfKBBevel

Description
The Bevel property is published by all available surface classes except TmfKBPageControl since the bevel for a page control is defined by the individual pages (TmfKBPage objects).

7.1.1.1.2 TmfKBSurface.Color

Sets the background color of the surface object.

Color: -$7FFFFFFF-1..$7FFFFFFF

Description
The Color property is a numeric value describing a color. The value is usually given in hexadecimal to make the number easier to read. In addition you can use the standard colors available in the property editor in the Keyboard Designer instead of assigning a numeric value. When assigning a standard color from script enclose the color name in double quotes, for example "clWhite".

The hexadecimal value for the Color property is built using three values, each specifying the amount of red, green and blue to apply to the color. The value $000000FF represents pure red,

© 2002 MountFocus Information Systems

$0000FF00 represents pure green and $00FF0000 represents pure blue. No color, $00000000, becomes black and "all colors", $00FFFFFF, becomes white.

Likewise, you can create mixes like $00FFFF00 (full green and full blue) to create cyan or $0000FFFF (full red and full green) to create yellow. This gives you approximately 16.7 million different combinations for the Color property.

Script values
The color property is numeric, so you can read and write the numeric value for a color in your scripts. However, you can read and write the standard colors by name (as a string).

Example
```
Sub SetColors()
  ' Set the color for Key1 to green
  Kbd.Keyboard.Key1.Color = 65280 ' 65280 = $0000FF00

  ' Set the color for Key2 to the standard active caption color
  Kbd.Keyboard.Key2.Color = "clActiveCaption"
End Sub
```

A quick guide to hexadecimal
Hexadecimal numbers are often used in programming, and they are not so different from the numbers you are used to. In our normal way of representing numbers, we use decimal notation, which is **based** on the value 10 - that is there are 10 digits ranging from 0 through 9. When we have a value that can not be represented with a single digit, we add another digit which is multiplied by the base (10). The decimal value for 9 + 1 is 10, or "one times base plus zero".

In hexadecimal the **base** is 16 - that is there are 16 digits ranging from $0 through $F (0123456789ABCDEF). (To separate hexadecimal values from decimal values, hexadecimal values are prefixed with a dollar sign.) So in hexadecimal, the value for 9 + 1 is $A - still represented by a single digit. Otherwise the same rules apply to hexadecimal. The value 9 + 7 is written as $10 in hexadecimal - that is "one times base plus zero".

What's the point? The computer stores all values in bytes. A byte can hold a value from 0 to 255 (decimal) or $0 to $FF in hexadecimal. If you do the math on your calculator you'll find that $FF represents $F (15) times base (16) plus $F (15) = 255.

From this you can see that the color property above is defined by three bytes (actually four, but we only use three of them) represented by six digits in hexadecimal. Its a lot easier to see that $00FF0000 represents pure blue (or at least pure something) than to come to the same conclusion from the decimal value (16711680 in this case).

There are two ways to quickly find color values without doing all the math. One is to use the Set color option available from the popup menu in the Keyboard Designer (right click the object you want to set the color for). If you still want to experiment with color codes, you can use the calculator applet that comes with Windows to do conversion to and from hexadecimal.

7.1.1.1.3 TmfKBSurface.pLeft

Sets the left position of the surface.

pLeft: integer

Description
The pLeft property sets the position of the left edge of the surface relative to its parent (the object it is placed on). This property is only available for descendants of the TmfKBMoveableSurface class. Note that pLeft is zero based, so the leftmost position for a surface has a pLeft value of 0.

© 2002 MountFocus Information Systems

7.1.1.1.4 TmfKBSurface.pTop

Sets the top position of the surface.

pTop: integer

Description
The pTop property sets the position of the top edge of the surface relative to its parent (the object it is placed on). This property is only available for descendants of the TmfKBMoveableSurface class. Note that pTop is zero based, so the topmost position for a surface has a pTop value of 0.

7.1.1.1.5 TmfKBSurface.sHeight

Sets the vertical size of the surface.

sHeight: integer

Description
The sHeight property sets the vertical size of the surface in pixels. This property is only available for descendants of the TmfKBSizeableSurface class.

7.1.1.1.6 TmfKBSurface.sWidth

Sets the horizontal size of the surface.

sWidth: integer

Description
The sWidth property sets the horizontal size of the surface in pixels. This property is only available for descendants of the TmfKBSizeableSurface class.

7.1.1.2 TmfKBSurfaceEvents

Events in *italics* are unavailable and can only be used with descendant classes that publish them.

OnButtonDown
OnButtonUp
OnMouseEnter
OnMouseLeave
OnPointerMove

7.1.1.2.1 TmfKBSurface.OnButtonDown

Occurs when the left mouse button is pressed.

OnButtonDown: procedure(Sender, X, Y)

Description
The OnButtonDown event occurs when the mouse pointer is over the surface object and the left mouse button is pressed.

Parameters
- Sender contains a reference to the surface object that received the event.
- X contains the horizontal position of the mouse pointer at the time of the event, relative to the upper left corner of the surface.
- Y contains the vertical position of the mouse pointer at the time of the event, relative to the upper

© 2002 MountFocus Information Systems

left corner of the surface.

7.1.1.2.2 TmfKBSurface.OnButtonUp

Occurs when the left mouse button is released.

OnButtonUp: procedure(Sender, X, Y)

Description
The OnButtonUp event occurs when the mouse pointer is over the surface object and the left mouse button is released.

Parameters
- Sender contains a reference to the surface object that received the event.
- X contains the horizontal position of the mouse pointer at the time of the event, relative to the upper left corner of the surface.
- Y contains the vertical position of the mouse pointer at the time of the event, relative to the upper left corner of the surface.

7.1.1.2.3 TmfKBSurface.OnMouseEnter

Occurs when the mouse pointer enters the surface.

OnMouseEnter: procedure(Sender)

Description
The OnMouseEnter event occurs when the mouse pointer is moved from another control to the surface object.

Parameters
- Sender contains a reference to the surface object that received the event.

7.1.1.2.4 TmfKBSurface.OnMouseLeave

Occurs when the mouse pointer leaves the surface.

OnMouseLeave: procedure(Sender)

Description
The OnMouseLeave event occurs when the mouse pointer is moved from the surface object to another control.

Parameters
- Sender contains a reference to the surface object that received the event.

© 2002 MountFocus Information Systems

7.1.1.2.5 TmfKBSurface.OnPointerMove

Occurs when the mouse pointer moves over the surface.

OnPointerMove: procedure(Sender, X, Y)

Description
The OnPointerMove event occurs when the mouse pointer moves over the surface. The event occurs regardless of the state of the mouse button (pressed or not).

Parameters
- Sender contains a reference to the surface object that received the event.
- X contains the horizontal position of the mouse pointer at the time of the event, relative to the upper left corner of the surface.
- Y contains the vertical position of the mouse pointer at the time of the event, relative to the upper left corner of the surface.

7.1.1.3 TmfKBPage

A TmfKBPage object represents a single page on a TmfKBPageControl object. A page cannot be sized or moved, since its size and position is determined by the page control object.

7.1.1.3.1 TmfKBPageProperties

Bevel
Color

7.1.1.3.2 TmfKBPageEvents

OnActivate
OnButtonDown
OnButtonUp
OnDeactivate
OnMouseEnter
OnMouseLeave
OnPointerMove

7.1.1.3.2.1 TmfKBPage.OnActivate

Occurs when the page is activated.

OnActivate: procedure(Sender)

Description
A page control object can contain many pages, but only one can be active (visible) at a time. When a page change occurs, the OnActivate event is triggered for the page that becomes active. See TmfKBPageControl for a list of events that occur during a page change.

Parameters
- Sender contains a reference to the page object that received the event.

© 2002 MountFocus Information Systems

7.1.1.3.2.2 TmfKBPage.OnDeactivate

Occurs when the page is deactivated.

OnDeactivate: procedure(Sender)

Description
A page control object can contain many pages, but only one can be active (visible) at a time. When a page is active and the page control changes to a different page, the OnDeactivate event is triggered. See TmfKBPageControl for a list of events that occur during a page change.

Parameters
- Sender contains a reference to the page object that received the event.

7.1.1.3.3 TmfKBPageMethods

Methods are only available from script.

Activate

7.1.1.3.3.1 TmfKBPage.Activate

Activates the page.

Activate

Description
The Activate method of the TmfKBPage class activates the page. That is, it signals the TmfKBPageControl which owns the page to make the page active.

Example
```
Sub ShowActivate()
  Dim Pg

  Set Pg = Kbd.GetControl( "Page1" )
  Pg.Activate
End Sub
```

7.1.1.4 TmfKBSizeableSurface

The TmfKBSizeableSurface introduces the ability to change size. It does so by publishing the sHeight and sWidth properties of the TmfKBSurface class.

7.1.1.4.1 TmfKBSizeableSurfaceProperites

Properties in *italics* are unavailable and can only be used with descendant classes that publish them.

Bevel
Color
pLeft
pTop
sHeight
sWidth

© 2002 MountFocus Information Systems

7.1.1.4.2 TmfKBKeyboard

Every keyboard file has one, and only one, keyboard object. The main tasks of the keyboard object is to provide a surface for other objects to be placed on, to define the size of the Runtime Keyboard window and to be a placeholder for global settings. Note that you cannot move the keyboard object. It is always positioned in the upper left corner of the Runtime Keyboard window. You can, of course, move the Runtime Keyboard window.

7.1.1.4.2.1 TmfKBKeyboardProperties

Properties marked **s** are only available from script during runtime, and thus do not appear in the Keyboard Designer.

Bevel
Color
GlobalPageControl
GlobalPageCount **s**
GlobalPageIndex **s**
GlobalPageName **s**
GridHeight
GridWidth
IndStates
KeySound
ScriptLanguage
sHeight
sWidth
TimerEnabled
TimerInterval

Reference to the page control object that will respond to version 2.x COM calls.

GlobalPageControl: TmfKBPageControl

Description
Since earlier versions of the Runtime Keyboard only provided one set of pages (i.e. one "page control"), the COM functions to perform page changes did not need to specify which page control they wanted to operate on. To provide compatibility with programs that use these COM functions, this property can be set to one of the page control objects on the keyboard. The effect is that previous version COM calls will be redirected to the GlobalPageControl.

Returns the number of pages in the GlobalPageControl.

GlobalPageCount: integer

Description
Read this property to find the number of page objects contained in the GlobalPageControl. If no page control is assigned to the GlobalPageControl property, the value returned is 0.

This is a read only property and it is only available from script.

Get or set the active page in the GlobalPageControl by its index.

GlobalPageIndex: integer

© 2002 MountFocus Information Systems

Description
You can read this property to obtain the index of the currently active page in the GlobalPageControl. If no page is active, or the GlobalPageControl property is not assigned to a page control object, the returned value is -1. Otherwise, the value returned ranges from 0 to GlobalPageCount -1. So, the first page has an index of 0, the second page an index of 1 and so on. You can also set this property to any value in the same range. Setting the property to an illegal value causes the GlobalPageControl to display no page.

This property is only available from script.

For simplicity, it is recommended that you use the GlobalPageName property instead of this.

Get or set the active page in the GlobalPageControl by its name.

GlobalPageName: string

Description
You can read this property to obtain the name of the currently active page in the GlobalPageControl. If no page is active, or the GlobalPageControl property is not assigned to a page control object, the returned value is an empty string. You can also set this property to the name of a page object contained in the GlobalPageControl. If the name you assign is invalid, no page change occurs.

This property is only available from script.

Defines the height of the grid used in the Keyboard Designer.

GridHeight: integer

Description
You can use this property to set the vertical size of the grid used during keyboard design. The grid height setting affects all surface objects on the keyboard. It is stored in the keyboard file, but it is ignored by the Runtime Keyboard. Changing this property during runtime has no effect.

Defines the width of the grid used in the Keyboard Designer.

GridWidth: integer

Description
You can use this property to set the horizontal size of the grid used during keyboard design. The grid width setting affects all surface objects on the keyboard. It is stored in the keyboard file, but it is ignored by the Runtime Keyboard. Changing this property during runtime has no effect.

Contains a list of indicator state objects (TmfKBIndState) for the keyboard.

IndStates: TmfKBIndStateList

Description
To edit the list of indicator state object in the Keyboard Designer, click the ellipsis (...) button in the property editor. To access the indicator state objects from script use the GetIndState method.

© 2002 MountFocus Information Systems

Contains a sound that is played when a key is pressed.

KeySound: TmfKBSound

Description
You can load a sound file (.wav) into this property to have the Runtime Keyboard play that sound whenever a key is pressed. The sound file is copied into the keyboard file so there is no need to distribute the sound file together with the keyboard file. The Runtime Keyboard has a setting to switch this feature off even if a sound is loaded into this property. Click the ellipsis (...) button in the property editor to load a sound file into this property, or to clear the loaded sound.

Select the script language to use for this keyboard.

ScriptLanguage: string

Description
You can choose which language to use for writing scripts in your keyboards. To change the script language simply enter the name of the script language you want to use in the ScriptLanguage property. Before you do so, you should make sure that the definition of the language has been set up in the Options dialog.

Important. The languages available may differ from system to system. Make sure that the language you use is available on all systems where you intend to run your keyboard.

Enable or disable the execution of the OnTimer event.

TimerEnabled: boolean

Description
When the TimerEnabled property is set to True and the TimerInterval property has a value greater than 0, the Runtime Keyboard will trigger the OnTimer event at the specified interval. If the TimerEnabled property is set to False or the TimerInterval property is 0, the OnTimer event is not triggered.

Set the interval between each time the OnTimer event is triggered.

TimerInterval: integer

Description
Use this property to set the amount of time between each time the OnTimer event is triggered. The interval is specified in milliseconds (1000ms = 1 second). Due to the nature of the timer the OnTimer event can not be triggered more than about 18 times per second - or every 55ms (approximately). Using values less than 55 will not cause the OnTimer event to be called more often. For the OnTimer event to be triggered, the TimerInterval must be greater than 0 and TimerEnabled property must be set to True.

© 2002 MountFocus Information Systems

7.1.1.4.2.2 TmfKBKeyboardEvents

OnButtonDown
OnButtonUp
OnHide
OnLoad
OnMouseEnter
OnMouseLeave
OnPointerMove
OnShow
OnTimer

Occurs when the Runtime Keyboard is about to hide.

OnHide: procedure(Sender)

Description
The OnHide event is triggered by the Runtime Keyboard before the keyboard is hidden. The OnHide event is triggered regardless of how the Runtime Keyboard received the command to hide. I.e. it is triggered when the user selects the Hide command from the system tray menu, when the Hide command is executed from script, and when the Hide COM function is called.

Parameters
- Sender contains a reference to the keyboard object.

Occurs when the Runtime Keyboard has loaded a keyboard file.

OnLoad: procedure(Sender)

Description
The OnLoad event is triggered by the Runtime Keyboard after a keyboard file has been loaded and all objects are created. Note that the OnLoad event is triggered regardless of how a new keyboard file was introduced to the Runtime Keyboard. I.e. it will be triggered when the Runtime Keyboard is started with a command line parameter specifying the file to load, when the Runtime Keyboard is started with the -o command line parameter and the user selects a keyboard file, when the LoadKeyboard COM function is used, and when the KeyboardFile, KeyboardAsBinary or KeyboardAsText COM properties are set (written to).

Parameters
- Sender contains a reference to the keyboard object.

Occurs when the Runtime Keyboard becomes visible.

OnShow: procedure(Sender)

Description
The OnShow event is triggered by the Runtime Keyboard after the keyboard is made visible by a Show command. The OnShow event is triggered regardless of how the Runtime Keyboard received the command to show. I.e. it is triggered when the user selects the Show command from the system tray menu, when the Show command is executed from script, and when the Show COM function is called.

© 2002 MountFocus Information Systems

Parameters
•Sender contains a reference to the keyboard object.

Occurs at the interval specified in the TimerInterval property.

OnTimer: procedure(Sender)

Description
When the TimerInterval property is greater than 0 and the TimerEnabled property is set to True, the OnTimer event will occur at the specified interval.

Parameters
•Sender contains a reference to the keyboard object.

7.1.1.4.2.3 TmfKBKeyboardMethods

Methods are only available from script.

GetControl
GetIndState

Returns a reference to an object on the keyboard.

GetControl(Name: string): TmfKBSurface
GetControl(Name: string): TmfKBControl

Description
The GetControl function searches all surfaces and controls on the keyboard and returns a reference to an object with a name matching the string specified in the Name parameter. Such an object may be a TmfKBSurface object (keyboard, page control, panel etc.) or a TmfKBControl object (key, label, region etc.). If no object is found the function fails (raises an error). The call to GetControl should therefore be written in a try..except/try..catch block (in VBScript use "On Error Resume Next").

Example
```
Sub KeyboardLoad( Sender )
  Dim Pg

  ' Sender contains a reference to the keyboard object
  On Error Resume Next
  Set Pg = Sender.GetControl( "Page1" )
  Pg.Activate
End Sub
```

Returns a reference to an indicator state object.

GetIndState(IndNo: integer): TmfKBIndState
GetIndState(IndName: string): TmfKBIndState

Description
The GetIndState function searches the defined indicator states for the keyboard for an indicator state that has a matching IndNo property or Name. The GetIndState function works with both the built in indicator states for CapsLock, NumLock and ScrollLock as well as any user defined indicator states.

© 2002 MountFocus Information Systems

Example

```
Sub CheckIndicators()
  Dim Ind

  On Error Resume Next

  Set Ind = Kbd.Keyboard.GetIndState( "CapsLock" )
  If Not IsObject( Ind ) Then
    ' CapsLock is not defined in this keyboard
  End If

  Set Ind = Kbd.Keyboard.GetIndState( "NumLock" )
  If Not IsObject( Ind ) Then
    ' NumLock is not defined in this keyboard
  End If

  ' The ScrollLock indicator state has an IndNo value of -3
  Set Ind = Kbd.Keyboard.GetIndState( -3 )
  If Not IsObject( Ind ) Then
    ' ScrollLock is not defined in this keyboard
  End If
End Sub
```

7.1.1.4.3 TmfKBMoveableSurface

The TmfKBMoveableSurface class introduces the ability to change the position of the surface. It does so by publishing the pLeft and pTop properties of the TmfKBSurface class.

7.1.1.4.3.1 TmfKBMoveableSurfaceProperties

Properties in *italics* are unavailable and can only be used with descendant classes that publish them.

Bevel
Color
pLeft
pTop
sHeight
sWidth

7.1.1.4.3.2 TmfKBPanel

Panel objects are intended for situations where you want to group some objects and optionally surround the group with a 3D border or different background color. Moving a panel object will also move all its child objects. The TmfKBPanel class publishes all properties and events defined in its ancestor classes.

Bevel
Color
pLeft
pTop
sHeight
sWidth

© 2002 MountFocus Information Systems

OnButtonDown
OnButtonUp
OnMouseEnter
OnMouseLeave
OnPointerMove

7.1.1.4.3.3 TmfKBPageControl

Page control objects are a special kind of surface objects that can not have any child controls except page objects (TmfKBPage). All pages contained within a page control object are sized and positioned according to the size and position of the page control object. Only one page may be visible at any given time. The Bevel and Color properties, as well as all the events, defined in the TmfKBSurface class are not available for page control objects. Instead you have access to these via the TmfKBPage class.

At runtime, the page control object keeps track of all page changes that occur and provides the ability to backtrack via the ClosePage function.

When a page change occurs the following events are triggered in the order listed.

- TmfKBPageControl.OnPageChanging
- TmfKBPage.OnDeactivate (for the page object active before the change)
- (Now the page change takes place)
- TmfKBPage.OnActivate (for the page object becoming active)
- TmfKBPageControl.OnPageChanged

Properties marked **s** are only available from script during runtime, and thus do not appear in the Keyboard Designer.

CurrentPage **s**
DefaultPage
PageCount **s**
PageIndex **s**
PageName **s**
pLeft
pTop
sHeight
sWidth

Refers to the page object currently active in the page control.

CurrentPage: TmfKBPage

Description
Read the CurrentPage property to obtain a reference to the currently active page in the page control object. If no page is active reading the property fails, so you should wrap a read of this property in a try..except/try..catch block (in VBScript use "On Error Resume Next"). You can also set the property to a valid reference to a page object to trigger a page change. The page object must exist as a child of the page control.

This property is only available from script. See TmfKBPageControl for a list of events that occur during a page change.

© 2002 MountFocus Information Systems

Refers to the page object activated when the keyboard is loaded.

DefaultPage: TmfKBPage

Description
The page control object uses this property to decide which of its pages should be activated when the keyboard is loaded. The property is automatically set to the first page object created for the page control, but you can change it to any valid page object. A valid page object means a page object which is contained by the page control (a child object of the page control). Changing the DefaultPage property at runtime has no effect.

Returns the number of pages in the page control.

PageCount: integer

Description
Read this property to find the number of page objects contained in the page control object.

This is a read only property and it is only available from script.

Get or set the active page in the page control by its index.

PageIndex: integer

Description
You can read this property to obtain the index of the currently active page in the page control. If no page is active the returned value is -1. Otherwise, the value returned ranges from 0 to PageCount - 1. So, the first page has an index of 0, the second page an index of 1 and so on. You can also set this property to any value in the same range. Setting the property to an illegal value causes the page control to display no page.

This property is only available from script. See TmfKBPageControl for a list of events that occur during a page change.

For simplicity, it is recommended that you use the PageName property instead of this.

Get or set the active page in the page control by its name.

PageName: string

Description
You can read this property to obtain the name of the currently active page in the page control. If no page is active the returned value is an empty string. You can also set this property to the name of a page object contained in the page control. If the name you assign is invalid, no page change occurs.

This property is only available from script. See TmfKBPageControl for a list of events that occur during a page change.

OnPageChanged
OnPageChanging

© 2002 MountFocus Information Systems

Occurs after a page change has finished.

OnPageChanged: procedure(Sender)

Description
The OnPageChanged event is the last event to be triggered when the page control changes the active page. See TmfKBPageControl for a list of events that occur during a page change.

Parameters
- Sender contains a reference to the page control object for which the page change occurred.

Occurs before a page change takes place.

OnPageChanging: procedure(Sender, Allow)

Description
The OnPageChanging event is the first event to be triggered when the page control changes the active page. It is also the only event that allows you to decide if the page change should be allowed or not. Note that the Allow parameter is an object with only one property - Value. To disallow the page change set Allow.Value to False, or to allow the page change set Allow.Value to True or simply leave it alone. See TmfKBPageControl for a list of events that occur during a page change.

Parameters
- Sender contains a reference to the page control object for which the page change is about to occur.
- Allow contains a reference to an object with one property - Value. Operate on the Value property, not the Allow parameter itself.

Methods are only available from script.

ClosePage

Closes the currently active page by activating the page that was previously active.

ClosePage

Description
When page changes occur, the page control object will keep track of these changes. Calling the ClosePage procedure will cause the page control to backtrack through its list of page changes (just like the Back function in your web browser). If no page changes have taken place in the page control, or if you have reached the end of the list of page changes, ClosePage does nothing.

Example
```
Sub BackToDefault( PageCtrl )
  While PageCtrl.CurrentPage <> PageCtrl.DefaultPage
    PageCtrl.ClosePage
  Wend
End Sub
```

© 2002 MountFocus Information Systems

7.2 Control classes

7.2.1 TmfKBControl

The TmfKBControl class is the ancestor for all control classes used by the keyboard. Features common for all controls are declared and implemented in this class, but not all are published (made available).

Control objects are keys (TmfKBKey), labels (TmfKBLabel), images (TmfKBImage), indicators (TmfKBIndicator) and regions (TmfKBRegion).

7.2.1.1 TmfKBControlProperties

Properties in *italics* are unavailable and can only be used with descendant classes that publish them.

Color
pLeft
pTop
sHeight
sWidth
Transparent

7.2.1.1.1 TmfKBControl.Color

Sets the background color of the control object.

Color: -$7FFFFFFF-1..$7FFFFFFF

Description
The Color property is a numeric value describing the a color. The value is usually given in hexadecimal to make the number easier to read. In addition you can use the standard colors available in the property editor in the Keyboard Designer instead of assigning a numeric value. When assigning a standard color from script enclose the color name in double quotes, for example "clWhite".

See TmfKBSurface.Color for details on assigning color values.

Although both the Color and Transparent properties are declared in TmfKBControl, they are used differently by the different control objects. Below is a brief explanation of these differences.

A key object (TmfKBKey) uses the color in the Color property to draw its background. But if you assign a background image in the Background property and set the Transparent property to False, the background image will overwrite the background color. This is because the Transparent property for a key determines the transparency of the background image - not the key itself.

A label object (TmfKBLabel) and an indicator object (TmfKBIndicator) uses the color in the Color property to draw its background if, and only if, its Transparent property is set to False. In other words, a transparent label or indicator has no background. To set the color of the text in a label object, use the TFont.Color property.

An image object (TmfKBImage) does not have a background color by default. It is either opaque (Transparent set to False) which causes the image to be drawn as is, or transparent (Transparent set to True) which causes only the non-transparent part of the image to be drawn and the background is left alone. In either case, no background color is used. So, to use the background color you must set the Transparent and the UseColor properties to True. This will cause the image

© 2002 MountFocus Information Systems

object to first fill its background with the selected color and then draw the image transparently.

7.2.1.1.2 TmfKBControl.pLeft

Sets the left position of the control.

pLeft: integer

Description
The pLeft property sets the position of the left edge of the control relative to its parent (the object it is placed on). This property is available for all control objects. Note that pLeft is zero based, so the leftmost position for a control has a pLeft value of 0.

7.2.1.1.3 TmfKBControl.pTop

Sets the top position of the control.

pTop: integer

Description
The pTop property sets the position of the top edge of the control relative to its parent (the object it is placed on). This property is available for all control objects. Note that pTop is zero based, so the topmost position for a control has a pTop value of 0.

7.2.1.1.4 TmfKBControl.sHeight

Sets the vertical size of the control.

sHeight: integer

Description
The sHeight property sets the vertical size of the control in pixels. This property is available for all control objects.

7.2.1.1.5 TmfKBControl.sWidth

Sets the horizontal size of the control.

sWidth: integer

Description
The sWidth property sets the horizontal size of the control in pixels. This property is available for all control objects.

7.2.1.1.6 TmfKBControl.Transparent

Sets the transparency for a control.

Transparent: boolean

Description
All control objects, except regions, publish the Transparent property. However, they use it slightly differently.

© 2002 MountFocus Information Systems

Keys (TmfKBKey) apply the Transparent property to the background image (Background property) if one is loaded. So, the key itself cannot be transparent, but the background image can.

Labels (TmfKBLabel) and indicators (TmfKBIndicator) will only draw their background if the Transparent property is set to False. When the Transparent property is set to True, a label object will only draw its text and an indicator object will only draw its caption objects.

Images (TmfKBImage) again apply the Transparent property to the bitmap (Image property) it contains.

Bitmaps and transparency
A bitmap is defined as a graphics object (file) loaded into the Background property of a key object, or the Image property of an image object. When you use the Transparent property with a bitmap, one of the colors in the bitmap is considered the transparent color. When the bitmap is drawn, all pixels in the transparent color are ignored (not drawn). The transparent color is always the color in the lower left corner of the bitmap.

7.2.1.2 TmfKBControlEvents

Events in *italics* are unavailable and can only be used with descendant classes that publish them.

OnButtonDown
OnButtonUp
OnMouseEnter
OnMouseLeave
OnPointerMove

7.2.1.2.1 TmfKBControl.OnButtonDown

Occurs when the left mouse button is pressed.

OnButtonDown: procedure(Sender, X, Y)

Description
The OnButtonDown event occurs when the mouse pointer is over the control object and the left mouse button is pressed.

Parameters
- Sender contains a reference to the control object that received the event.
- X contains the horizontal position of the mouse pointer at the time of the event, relative to the upper left corner of the control.
- Y contains the vertical position of the mouse pointer at the time of the event, relative to the upper left corner of the control.

7.2.1.2.2 TmfKBControl.OnButtonUp

Occurs when the left mouse button is released.

OnButtonUp: procedure(Sender, X, Y)

Description
The OnButtonUp event occurs when the mouse pointer is over the control object and the left mouse button is released.

© 2002 MountFocus Information Systems

Parameters
•Sender contains a reference to the control object that received the event.
•X contains the horizontal position of the mouse pointer at the time of the event, relative to the upper left corner of the control.
•Y contains the vertical position of the mouse pointer at the time of the event, relative to the upper left corner of the control.

7.2.1.2.3 TmfKBControl.OnMouseEnter

Occurs when the mouse pointer enters the control.

OnMouseEnter: procedure(Sender)

Description
The OnMouseEnter event occurs when the mouse pointer is moved from another object to the control object.

Parameters
- Sender contains a reference to the control object that received the event.

7.2.1.2.4 TmfKBControl.OnMouseLeave

Occurs when the mouse pointer leaves the control.

OnMouseLeave: procedure(Sender)

Description
The OnMouseLeave event occurs when the mouse pointer is moved from the control object to another object.

Parameters
- Sender contains a reference to the control object that received the event.

7.2.1.2.5 TmfKBControl.OnPointerMove

Occurs when the mouse pointer moves over the control.

OnPointerMove: procedure(Sender, X, Y)

Description
The OnPointerMove event occurs when the mouse pointer moves over the control. The event occurs regardless of the state of the mouse button (pressed or not).

Parameters
- Sender contains a reference to the control object that received the event.
- X contains the horizontal position of the mouse pointer at the time of the event, relative to the upper left corner of the control.
- Y contains the vertical position of the mouse pointer at the time of the event, relative to the upper left corner of the control.

© 2002 MountFocus Information Systems

7.2.1.3 TmfKBKey

The basic purpose of the key object is, of course, to send keystrokes to Windows. However, the key object can also perform a number of other tasks such as changing pages on a page control, allowing movement of the Runtime Keyboard or shutting down the Runtime Keyboard. In addition you can write scripts that handle the events of the key.

The following events occur, in the order listed, for the key object when the left mouse button is pressed and released over the key object:

- OnButtonDown
- OnKeyPressed
- OnButtonUp
- OnKeyReleased

7.2.1.3.1 TmfKBKeyProperties

Properties marked **s** are only available from script during runtime, and thus do not appear in the Keyboard Designer.

Background
Caption
CloseKeyboard
ClosePage
Color
KeyState
Keystrokes
KeyType
MoveKeyboard
pLeft
PressedBevel
PressedOffsetX
PressedOffsetY
pTop
ReleasedBevel
Repeating
SelectPage
sHeight
ShiftState **s**
sWidth
Transparent

7.2.1.3.1.1 TmfKBKey.Background

Holds a background image (bitmap) for the key.

Background: TPicture

Description
To load a bitmap image into the Background property of a key click the ellipsis (...) button in the property editor. The image is always stretched to the size of the key, minus the active bevel (see PressedBevel and ReleasedBevel). As an example, if the key is 40 by 40 pixels and the active bevel has its Outer property set to bvRaised and OuterWidth to 2, the image will be drawn at left position 2, top position 2 and 36 pixels wide and 36 pixels high. Set the Transparent property of the key to True to make the background image transparent.

© 2002 MountFocus Information Systems

7.2.1.3.1.2 TmfKBKey.Caption

Holds a list of controls that make out the caption of the key.

Caption: TmfKBCaptionList

Description
To access the caption objects for a key in the Keyboard Designer, click the ellipsis (...) button in the property editor or right click the key and choose "Edit caption" from the menu. To access the caption objects from script, either use the GetControl method or use dot notation (for example `Key1.Label1`).

7.2.1.3.1.3 TmfKBKey.CloseKeyboard

Causes the key to close the Runtime Keyboard when pressed.

CloseKeyboard: boolean

Description
When the CloseKeyboard property is set to True, pressing the key will cause it to shut down the Runtime Keyboard application. You should not use this feature with keyboards intended to be controlled from other applications via COM. When using COM, let the controlling application handle startup and shutdown of the Runtime Keyboard.

When CloseKeyboard is set to True, the ClosePage, MoveKeyboard, Repeating and SelectPage properties are all cleared.

7.2.1.3.1.4 TmfKBKey.ClosePage

Calls the ClosePage method of a page control object.

ClosePage: TmfKBPageControl

Description
Set the ClosePage property to a valid page control object. Doing so will cause the key to call the ClosePage method of the page control object when the key is pressed.

Setting the ClosePage property clears the CloseKeyboard, MoveKeyboard and Repeating properties.

7.2.1.3.1.5 TmfKBKey.KeyState

Identifies if the key is currently pressed or not.

KeyState: TKeyState

TKeyState = (ksReleased, ksPressed)

Description
The KeyState property holds the current state of a key. For a normal key (KeyType property set to ktNormal) the KeyState will be set to ksPressed for as long as the user presses the key (holds the mouse button pressed over the key). Otherwise KeyState is set to ksReleased. A shift key (KeyType property set to ktShift) will remain in the ksPressed state until it is used or pressed again. Shifted keys also use the ShiftState property to hold their shifted state.

Important: This property is NOT stored in your keyboard files. It is available in the Keyboard Designer only to let you see the visual representation of a key in its pressed state without having to

© 2002 MountFocus Information Systems

run the keyboard. All keys are initialized to ksReleased when loaded.

Script values
This property is treated as a string when used from script. Reading it will return either "ksReleased" or "ksPressed" and only the same two strings are accepted when writing to it.

7.2.1.3.1.6 TmfKBKey.Keystrokes

Holds the keystrokes associated with the key.

Keystrokes: TmfKBKeystrokes

Description
The Keystrokes property holds a reference to an object that contains the keystrokes associated with the key. Click the ellipsis (...) button in the property editor to open the keystrokes editor.

7.2.1.3.1.7 TmfKBKey.KeyType

Sets the key to either a normal key or a shifted key.

KeyType: TKeyType

TKeyType = (ktNormal, ktShift, ktShiftLock)

Description
A key with its KeyType set to ktNormal will simply send its keystrokes when pressed. A key with its KeyType set to ktShift will instead remain pressed to allow you to press another key to create a combined keystroke. For example, the <Ctrl> key has its KeyType set to ktShift so that you can press <Ctrl> first and then <C> (which is a normal key) to produce a <Ctrl>+<C> keystroke. Note that all shifted keys are automatically released after you press a normal key (you don't have to press them again to release them). A key with its KeyType set to ktShiftLock will remain pressed to allow you to press another key to create a combined keystroke and remain pressed until manually released.

The actual keyboard messages sent to Windows will vary depending on the Repeating property.

Script values
This property is treated as a string when used from script. Reading it will return "ktNormal", "ktShift" or "ktShiftLock" and only the same three strings are accepted when writing to it.

7.2.1.3.1.8 TmfKBKey.MoveKeyboard

Sets the Runtime Keyboard Position property to kpAnywhere which allows the user to move the keyboard once.

MoveKeyboard: boolean

Description
When the MoveKeyboard property is set to True and the key is pressed, the Runtime Keyboard goes into "move once" mode. That is, the user is allowed to drag the keyboard to a new location but when he ends the drag operation, the keyboard is stationary.

© 2002 MountFocus Information Systems

7.2.1.3.1.9 TmfKBKey.PressedBevel

Defines the 3D border around the key when it is pressed.

PressedBevel: TmfKBBevel

Description
The PressedBevel property is used when the key is pressed - that is, as long as the KeyState is ksPressed. Use this property in combination with the PressedOffsetX|Y properties and the ReleasedBevel property to define the 3D look and feel of a key.

7.2.1.3.1.10 TmfKBKey.PressedOffsetX|Y

Defines the movement of a key's caption when pressed.

PressedOffsetX: integer
PressedOffsetY: integer

Description
These two properties define the horizontal (PressedOffsetX) and vertical (PressedOffsetY) movement of the objects on a key's caption when the key is pressed. They are used in combination with the PressedBevel and ReleasedBevel properties to give a proper 3D look to a key. As an example, most of the keys defined in the keypool have a raised 3D border when released and a lowered 3D border when pressed, both with a border width of 2 pixels. They also have their PressedOffsetX and PressedOffsetY properties set to 2, which causes all caption objects (usually a label object) to be moved 2 pixels to the right and 2 pixels down when the key is pressed.

Note that positive values in these properties moves objects to the right and down, while negative values move in the opposite directions.

7.2.1.3.1.11 TmfKBKey.ReleasedBevel

Defines the 3D border around the key when it is released.

ReleasedBevel: TmfKBBevel

Description
The ReleasedBevel property is used when the key is released - that is, as long as the KeyState is ksReleased. Use this property in combination with the PressedOffsetX|Y properties and the PressedBevel property to define the 3D look and feel of a key.

7.2.1.3.1.12 TmfKBKey.Repeating

Makes the key repeat its keystrokes while pressed.

Repeating: boolean

Description
Setting the Repeating property to True causes the key to act exactly like a key on your physical keyboard. It produces its keystroke when pressed, and continue to do so until released. The disadvantage of a repeating key is that it can only produce a single keystroke. If you define more than one virtual key in the Keystrokes property, only the first will be used by the key.

© 2002 MountFocus Information Systems

These tables show, in detail, when keyboard messages are produced and sent depending on the setting in the Repeating, KeyType and ShiftState properties.

Repeating = False	When pressed (mouse button down)	When released (mouse button up)	
KeyType = ktNormal	No action taken	KeyDown messages are produced for all keystrokes in the shift queue, then key messages as defined in this key's Keystrokes property are produced (both KeyDown and KeyUp) and finally KeyUp messages for all keystrokes in the shift queue are produced. Finally all the keystrokes produced are sent to Windows in one block.	
KeyType = ktShift	No action taken	If ShiftState is ksReleased, the keystrokes for the key are added to the shift queue and the ShiftState property is set to ksPressed.	If ShiftState is ksPressed, the keystrokes for the key are removed from the shift queue and the ShiftState property is set to ksReleased.

Repeating = True	When pressed (mouse button down)		When released (mouse button up)	
KeyType = ktNormal	KeyDown messages for all keystrokes in the shift queue are produced and sent to Windows. Then, a KeyDown message for the keystroke defined for this key is sent to Windows. After a short pause, this key's KeyDown message is repeatedly sent to Windows until the key is released or another key is pressed.		A KeyUp message for this key's keystroke is sent to Windows. Finally, KeyUp messages for all keystrokes in the shift queue are produced and sent to Windows.	
KeyType = ktShift	If ShiftState is ksReleased, a KeyDown message is sent to Windows. This message is repeatedly sent to Windows until the key is released or another key is pressed.	If ShiftState is ksReleased, a KeyUp message is sent to Windows.	If ShiftState is ksReleased it is set to ksPressed.	If ShiftState is ksPressed, it is set to ksReleased.

The best way to experiment with these settings and their results is to create a few keys with different settings and use them with the Keyboard Tester utility program that was installed with the Keyboard Designer. You can also use the Keyboard Tester program to compare messages from your virtual keyboards to messages from the physical keyboard.

© 2002 MountFocus Information Systems

7.2.1.3.1.13 TmfKBKey.SelectPage

Makes the key cause a page change in a page control.

SelectPage: TmfKBPage

Description
You can set the SelectPage to refer to any page on any page control. Doing so will cause the key to call the Activate method of the page object when the key is pressed.

Setting the SelectPage property clears the CloseKeyboard and MoveKeyboard properties.

7.2.1.3.1.14 TmfKBKey.ShiftState

Holds the current shift state for a shifted key.

ShiftState: TKeyState

TKeyState = (ksReleased, ksPressed)

Description
A key that has its KeyType property set to ktShift is called a shifted key. It differs from other keys in that it remains pressed until used or pressed again. The ShiftState property holds the current state of a shifted key - pressed or released. Its value (state) follows the KeyState property after the mouse button has been pressed and released over the key (after a completed keystroke).

This property is only available from script.

Script values
This property is treated as a string when used from script. Reading it will return either "ksReleased" or "ksPressed" and only the same two strings are accepted when writing to it.

7.2.1.3.2 TmfKBKeyEvents

OnButtonDown
OnButtonUp
OnKeyPressed
OnKeyReleased
OnMouseEnter
OnMouseLeave
OnPointerMove

7.2.1.3.2.1 TmfKBKey.OnKeyPressed

Occurs when the key is pressed.

OnKeyPressed: procedure(Sender)

Description
This event occurs when the key is pressed, that is when the KeyState property is set to ksPressed. If the key has its Repeating property set to True, the initial KeyDown message is sent to Windows before this event occurs.

Parameters
- Sender contains a reference to the key object that received the event.

© 2002 MountFocus Information Systems

7.2.1.3.2.2 TmfKBKey.OnKeyReleased

Occurs when the key is released.

OnKeyReleased: procedure(Sender)

Description
This event occurs when the key is released, that is when the KeyState property is set to ksReleased. If the key has its Repeating property set to True, the final KeyUp message is sent to Windows before this event occurs.

Parameters
- Sender contains a reference to the key object that received the event.

7.2.1.4 TmfKBRegion

A region object defines an invisible area on the keyboard, so you can use regions to create a keyboard that is non-rectangular. Note that you can place a region object on a page. If you do, the region will only be active (create an invisible area) when the page is active.

The region object has no events.

7.2.1.4.1 TmfKBRegionProperties

pLeft
pTop
sHeight
sWidth

7.2.1.5 TmfKBStaticControl

The TmfKBStaticControl class is the ancestor for all controls that can be placed on a caption (the visual part of a key or an indicator). This class also publishes some properties and events that were defined in TmfKBControl.

Important: When a TmfKBStaticControl descendant is placed on a caption, it does not receive events. Mouse related events are sent to the owner (the key or indicator it is placed on) so these events have no purpose when the TmfKBStaticControl descendant is on a caption. The OnStateChanging and OnStateChanged events of the indicator object are not triggered either. To handle these events, create an indicator state object (TmfKBIndState) where you can handle the events and let the indicator object monitor the indicator state object.

7.2.1.5.1 TmfKBStaticControlProperties

Properties in *italics* are unavailable and can only be used with descendant classes that publish them.

Color
pLeft
pTop
sHeight
sWidth
Transparent

© 2002 MountFocus Information Systems

7.2.1.5.2 TmfKBStaticControlEvents

OnButtonDown
OnButtonUp
OnMouseEnter
OnMouseLeave
OnPointerMove

7.2.1.5.3 TmfKBIndicator

An indicator object holds two separate captions and switches between them based on its State property. When the State property is set to True the indicator object displays its OnCaption, and when the State property is False it displays its OffCaption. You can set the State property from script or you can use the MonitorIndicator property to link the indicator object to an indicator state object (TmfKBIndState).

Important: When an indicator is placed on the caption of a key or another indicator, its events are not triggered. All mouse related events are sent to the owner (the key or indicator it is placed on). To handle the OnStateChanging and OnStateChanged events, create an indicator state object to handle the events and set the indicator object's MonitorIndicator property to the name of this indicator state object.

7.2.1.5.3.1 TmfKBIndicatorProperties

Color
MonitorIndicator
OffBevel
OffCaption
OnBevel
OnCaption
pLeft
pTop
sHeight
State
sWidth
Transparent

Holds the name of the indicator state object (TmfKBIndState) to monitor.

MonitorIndicator: string

Description
When you set the MonitorIndicator property to the name of an indicator state object, the indicator object will automatically change its state according to the state of the indicator state object. For example, you can set the MonitorIndicator property to "CapsLock" to have the indicator change state when Caps Lock is turned on or off. When this property is left blank, you have to change the state of the indicator object yourself via script.

Defines the 3D border around the indicator when it is in its off state (State property set to False).

OffBevel: TmfKBBevel

Description
The OffBevel is used to draw the indicator's border as long as the State property is set to False.

© 2002 MountFocus Information Systems

Use this property in combination with the OnBevel property to create 3D effects for the indicator object.

Holds a list of controls that make up the caption of the indicator object when its State property is False.

OffCaption: TmfKBCaptionList

Description
To access the caption objects in the Keyboard Designer, click the ellipsis (...) button in the property editor or right click the indicator and choose "Edit off caption" from the menu. To access the caption objects from script, either use the GetControl method or use dot notation (for example `Indicator1.Label1`).

Defines the 3D border around the indicator when it is in its on state (State property set to True).

OnBevel: TmfKBBevel

Description
The OnBevel is used to draw the indicator's border as long as the State property is set to True. Use this property in combination with the OffBevel property to create 3D effects for the indicator object.

Holds a list of controls that make up the caption of the indicator object when its State property is True.

OnCaption: TmfKBCaptionList

Description
To access the caption objects in the Keyboard Designer, click the ellipsis (...) button in the property editor or right click the indicator and choose "Edit on caption" from the menu. To access the caption objects from script, either use the GetControl method or use dot notation (for example `Indicator1.Label1`).

Holds the state of the indicator object.

State: boolean

Description
The state of an indicator object is either True (On) or False (Off). When the State property is set to True, the indicator object uses the OnBevel and OnCaption properties to draw itself, otherwise it uses the OffBevel and OffCaption properties. To change the State of an indicator object at runtime, set this property via script or use the MonitorIndicator property to allow an indicator state object to set the State of the indicator object automatically.

Important: This property is NOT stored in your keyboard files. It is available in the Keyboard Designer only to let you see the visual representation of an indicator in its on and off states without having to run the keyboard. All indicators are initialized to Off (State = False) when loaded.

© 2002 MountFocus Information Systems

7.2.1.5.3.2 TmfKBIndicatorEvents

OnButtonDown
OnButtonUp
OnMouseEnter
OnMouseLeave
OnPointerMove
OnStateChanged
OnStateChanging

Occurs after the indicator object has changed its state.

OnStateChanged: procedure(Sender)

Description
Use this event to perform actions after the indicator object has changed its state. The new state of the indicator object is available from the State property.

Parameters
- Sender contains a reference to the indicator object that received the event.

Occurs before the indicator object changes its state.

OnStateChanging: procedure(Sender, Allow)

Description
Before an indicator object changes its state, it triggers this event to give you a last chance to disallow the change. The Allow parameter is an object with only one property - Value. To refuse the indicator to change states, set the Allow.Value property to False. To allow the state change, set Allow.Value to True or simply leave it alone. Note that refusing an indicator which monitors any of the three standard indicator state objects (CapsLock, NumLock and ScrollLock) to change states only affects the indicator object itself.

Parameters
- Sender contains a reference to the indicator object for which the state change is about to occur.
- Allow contains a reference to an object with one property - Value. Operate on the Value property, not the Allow parameter itself.

7.2.1.5.4 TmfKBLabel

Label objects are used to display text. Labels can have multiple Lines of Text and can be displayed using any available Font. It is recommended that you use standard fonts (fonts that come with Windows) since only the name of the font is stored in the keyboard file. If the font you use is not available on a system running your keyboard, Windows will replace it with another font and your keyboard may look very different from what you intended.

Important: When a label is placed on the caption of a key or an indicator, its events are not triggered since all mouse related events are sent to the owner (the key or indicator it is placed on).

© 2002 MountFocus Information Systems

7.2.1.5.4.1 TmfKBLabelProperties

AutoSize
Bevel
Color
Font
Lines
pLeft
pTop
sHeight
sWidth
Text
Transparent

Causes the label to change size according to the text and font automatically.

AutoSize: boolean

Description
When the AutoSize property is set to True, the label object will change its size (sWidth and sHeight properties) automatically whenever changes occur to the Text, Lines, Font or Bevel properties. The size set is calculated so that all the text will fit inside the border described by the Bevel property. Setting AutoSize to False causes the label to maintain its size regardless of changes to the other properties. This may cause the text to be clipped if it is too big to fit inside of the label.

Defines the 3D border around the label.

Bevel: TmfKBBevel

Description
Use the Bevel property to define a 3D border around the label object.

Defines the font to use to draw the label.

Font: TFont

Description
You can use any font available in Windows to draw the label. However, you should make sure that the system you intend to run the keyboard on has the same font installed. Otherwise Windows will change the actual font used to a "best guess" (which is not always very good) replacement.

Holds all the lines of text for the label.

Lines: TStringList

Description
A label object may have multiple lines of text. You can use the Lines property to handle all these lines, or you can use the Text property to handle only the first line of text. The Lines property is currently only available in the Keyboard Designer.

Holds the first line of text for the label object.

Text: string

© 2002 MountFocus Information Systems

Description
Since most labels have only one line of text the label object publishes the Text property to make it easier to access the first line of the Lines property. Use this property to manipulate the text of a single line label.

Note: When you change the Text property, either in the Keyboard Designer or via script in the Runtime Keyboard, all the lines of text in the Lines property are replaced with the new text (one line) that you set in the Text property.

Important: This property is not stored in your keyboard files since it really only references the first line of the Lines property (which is stored).

7.2.1.5.5 TmfKBImage

Image objects are used to hold a bitmap file to allow you to display graphics on your keyboards. You can load image files in bitmap (.bmp), icon (.ico) or Windows Metafile (.wmf or .emf) formats, but bitmaps are recommended as they can be manipulated more easily. Icon files are transparent (regardless of the setting of the Transparent property) and they do not allow stretching (sizing of the graphics object). Metafiles can be sized, but cannot be transparent. Bitmaps can be sized and can be transparent or opaque.

All image files used are stored in the keyboard file. You do not need to distribute the image file along with your keyboard.

7.2.1.5.5.1 TmfKBImageProperties

AutoSize
Bevel
Color
Image
pLeft
pTop
sHeight
sWidth
Transparent
UseColor

Causes the image object to size itself according to the size of the loaded bitmap.

AutoSize: boolean

Description
Set the AutoSize property to True to let the image object size itself to the size of the loaded bitmap. When AutoSize is set to False, you can resize the image object and the loaded bitmap will be stretched to fit the image object.

Note: Icon files cannot be resized.

© 2002 MountFocus Information Systems

Defines the 3D border around the image.

Bevel: TmfKBBevel

Description
Use the Bevel property to define a 3D border around the image object.

Holds the bitmap file containing the picture to display.

Image: TPicture

Description
To load a bitmap file into the image object, click the ellipsis (...) button in the property editor. In the dialog that appears you can load a new file or clear the loaded one.

Decides if the background of a transparent image should be filled with the image objects color.

UseColor: boolean

Description
When an image object has both its Transparent and UseColor properties set to True it will use the color set in the Color property to draw its background. If the Transparent property is set to True and the UseColor property is set to False, the image will not draw any background and will therefore let any existing objects "behind" the image object show through the transparent areas of the image. UseColor is ignored if Transparent is set to False.

7.3 Non visual classes

7.3.1 TmfKBNonVisual

The TmfKBNonVisual is the ancestor class for all special purpose keyboard classes.

7.3.1.1 TmfKBKeystrokes

The TmfKBKeystrokes class is used by the TmfKBKey class to hold the keystrokes associated with the key. This class is currently only available from the Keyboard Designer by clicking the ellipsis (...) button for the Keystrokes property or by right clicking a key and choosing "Edit keystrokes".

7.3.1.2 TmfKBKeyNode

The TmfKBKeyNode class holds a single keystroke. It also holds references to the first child keystroke and its next sibling. This class can currently only be manipulated via the TmfKBKeystrokes class.

7.3.1.3 TmfKBSound

The TmfKBSound class holds a sound file (.wav) that is played each time a key is pressed. This class is used only by the TmfKBKeyboard class so only one sound can be loaded at any given time. The sound is stored in the keyboard file so there is no need to distribute the sound file with your keyboards.

The TmfKBSound class has no properties or events. You can load or clear the sound in the

© 2002 MountFocus Information Systems

Keyboard Designer by clicking the ellipsis (...) button for the KeySound property.

7.3.1.3.1 TmfKBSoundMethods

Methods are only available from script.

LoadFromFile
Play

7.3.1.3.1.1 TmfKBSound.LoadFromFile

Loads a new sound file. The sound file must be in .wav (wave) format.

LoadFromFile(Filename: string)

Description
Call LoadFromFile to load a new sound file into the TmfKBSound object. The existing sound, if any, is first cleared. The Filename parameter is a string which must contain the full path and filename for the sound file to be loaded.

Example
```
Sub SetSound()
  Kbd.Keyboard.KeySound.LoadFromFile( "C:\Program Files\MountFocus\Keyboard\Key.wav" )
End Sub
```

7.3.1.3.1.2 TmfKBSound.Play

Plays the loaded sound, if any.

Play

Description
Call Play to play the loaded sound. If no sound is loaded or if the Runtime Keyboard SoundOn setting is set to off, no sound is played.

Example
```
' Play the sound when SomeEvent occurs
Sub SomeEvent( Sender )
  Kbd.Keyboard.KeySound.Play
End Sub
```

7.3.1.4 TmfKBBevel

Most controls (windowed and non-windowed) can have one or more bevels. A bevel is a 3D frame that is drawn around the control. By changing the different properties of the bevel you can change the visual 3D effect of the control. The TmfKBBevel class has no events.

The two main parts of a bevel are the Inner and Outer frames. They both provide the same functionality and as their names suggest, one is drawn inside of the other. In addition there is a third frame, Border, which is drawn outside of everything else. Each of the frames can be set independently of the others - both in style and width (thickness), but they all use the same colors.

© 2002 MountFocus Information Systems

7.3.1.4.1 TmfKBBevelProperties

Border
BorderWidth
BottomColor
Inner
InnerWidth
Outer
OuterWidth
Spacing
TopColor

7.3.1.4.1.1 TmfKBBevel.Border

Defines the style of the border for the control.

Border: TBevelCut

TBevelCut = (bvNone, bvLowered, bvRaised, bvSpace)

Description
Use the Border property when you need an extra frame outside of the Inner and Outer frames. The width (thickness) of the frame is decided by the BorderWidth property.

TBevelCut value	Visual effect
bvNone	No visual effect. That is - no frame is drawn.
bvLowered	The frame is drawn to appear sunken. The TopColor color is used on the right and bottom sides, and the BottomColor color is used on the left and top sides.
bvRaised	The frame is drawn to appear raised. The TopColor color is used on the left and top sides, and the BottomColor color is used on the right and bottom sides.
bvSpace	The owner control's color is used on all four sides, thus creating no 3D effect, but instead a blank area.

Script values
This property is treated as a string when used from script. Reading it will return either "bvNone", "bvLowered", "bvRaised" or "bvSpace" and only the same strings are accepted when writing to it.

7.3.1.4.1.2 TmfKBBevel.BorderWidth

Defines the width (thickness) of the Border frame.

BorderWidth: integer

Description
Set the BorderWidth property to a value greater than zero to define the width of the Border frame in pixels. If the Border property is set to bvNone, BorderWidth has no effect.

© 2002 MountFocus Information Systems

7.3.1.4.1.3 TmfKBBevel.BottomColor

Sets the shadow color of the bevel object.

Color: -$7FFFFFFF-1..$7FFFFFFF

Description
To create a 3D effect, the bevel object works with a BottomColor (shadow) and a TopColor (highlight). The same two colors are used for all the frames of a bevel object as described for the Border property.

See TmfKBSurface.Color for details on Color values.

7.3.1.4.1.4 TmfKBBevel.Inner

Defines the style of the inner frame for the control.

Inner: TBevelCut

Description
Use the Inner property to set the 3D effect for the innermost frame around a control. The width (thickness) of the frame is decided by the InnerWidth property. See the Border property for an explanation of the different values available for this property.

7.3.1.4.1.5 TmfKBBevel.InnerWidth

Defines the width (thickness) of the Inner frame.

InnerWidth: integer

Description
Set the InnerWidth property to a value greater than zero to define the width of the Inner frame in pixels. If the Inner property is set to bvNone, InnerWidth has no effect.

7.3.1.4.1.6 TmfKBBevel.Outer

Defines the style of the outer frame for the control.

Outer: TBevelCut

Description
Use the Outer property to set the 3D effect for the outer frame around a control. The Outer frame will appear between the Inner and Border frames. The width (thickness) of the frame is decided by the OuterWidth property. See the Border property for an explanation of the different values available for this property.

7.3.1.4.1.7 TmfKBBevel.OuterWidth

Defines the width (thickness) of the Outer frame.

OuterWidth: integer

Description
Set the OuterWidth property to a value greater than zero to define the width of the Outer frame in

© 2002 MountFocus Information Systems

pixels. If the Outer property is set to bvNone, OuterWidth has no effect.

7.3.1.4.1.8 TmfKBBevel.Spacing

Sets the space between the Inner and Outer frames.

Spacing: integer

Description
You can consider the Spacing property to be an extra frame between the Inner and Outer frames. Set spacing to zero to switch it off, or to the width (thickness) of the spacing in pixels. This extra frame is always drawn as if it was set to bvSpace.

7.3.1.4.1.9 TmfKBBevel.TopColor

Sets the highlight color of the bevel object.

Color: -$7FFFFFFF-1..$7FFFFFFF

Description
To create a 3D effect, the bevel object works with a BottomColor (shadow) and a TopColor (highlight). The same two colors are used for all the frames of a bevel object as described for the Border property.

See TmfKBSurface.Color for details on Color values.

7.3.1.5 TmfKBIndState

The TmfKBIndState class holds the state of a user defined or system defined setting. An indicator state is like an invisible switch that can be either on or off. There are three system defined indicator states, CapsLock, NumLock and ScrollLock, that are all updated automatically. In addition you can create up to 250 indicator state objects yourself, but you will also have to update their state (on or off). You can change the state of an indicator from your own application by using the SetIndicator COM function, or by writing scripts in the keyboard. To define new indicator state objects, click the ellipsis (...) button for the IndStates property.

You can connect a visual indicator object (TmfKBIndicator) to an TmfKBIndState object through the MonitorIndicator property of the indicator object. If you do, the indicator object will automatically update its state according to the state of the indicator state object.

7.3.1.5.1 TmfKBIndStateProperties

IndNo
Name
State

© 2002 MountFocus Information Systems

7.3.1.5.1.1 TmfKBIndState.IndNo

A unique number for the indicator state object.

IndNo: integer

Description
The IndNo property is a unique number identifying the indicator state object. Values allowed are from -3 (minus three) to 250 inclusive. Negative values are used for the system defined indicator state objects, CapsLock (-1), NumLock (-2) and ScrollLock (-3). The Runtime Keyboard will monitor the state of these three lock keys and update the state of their respective indicator state objects automatically. For user defined indicator state objects, use IndNo values of 0 through 250.

7.3.1.5.1.2 TmfKBIndState.Name

Sets a descriptive name for the indicator state object.

Name: string

Description
Use this property to give the indicator state object a descriptive name. It is recommended, but not required, that you follow the naming conventions for control names - start with a letter and follow up with letters, numbers and/or underscore (_).

7.3.1.5.1.3 TmfKBIndState.State

Holds the state of the indicator state object.

State: boolean

Description
The state of an indicator state object is either True (On) or False (Off). You can change the state of an indicator state object via the SetIndicator COM function or via script. When the state of an indicator state object changes, it automatically updates the state of all visual indicator objects (TmfKBIndicator) that have their MonitorIndicator property set to monitor this indicator state object.

7.3.1.5.2 TmfKBIndStateEvents

OnStateChanged
OnStateChanging

7.3.1.5.2.1 TmfKBIndState.OnStateChanged

Occurs after the indicator state object has changed its state.

OnStateChanged: procedure(Sender)

Description
Use this event to perform actions after the indicator state object has changed its state. The new state of the indicator state object is available from the State property.

Parameters
- Sender contains a reference to the indicator state object that received the event.

© 2002 MountFocus Information Systems

7.3.1.5.2.2 TmfKBIndState.OnStateChanging

Occurs before the indicator state object changes its state.

OnStateChanging: procedure(Sender, Allow)

Description
Before an indicator state object changes its state, it triggers this event to give you a last chance to disallow the change. The Allow parameter is an object with only one property - Value. To refuse the indicator state object to change states, set the Allow.Value property to False. To allow the state change, set Allow.Value to True or simply leave it alone. Note that setting Allow.Value to False for any of the system defined indicator state objects (CapsLock, NumLock and ScrollLock) has no effect.

Parameters
- Sender contains a reference to the indicator state object for which the state change is about to occur.
- Allow contains a reference to an object with one property - Value. Operate on the Value property, not the Allow parameter itself.

7.3.1.6 TPicture

The TPicture class is a placeholder for bitmap files. You can load image files in bitmap (.bmp), icon (.ico) or Windows Metafile (.wmf or .emf) formats, but bitmaps are recommended as they can be manipulated more easily. Icon files are transparent (regardless of the setting of the Transparent property) and they do not allow stretching (sizing of the graphics object). Metafiles can be sized, but cannot be transparent. Bitmaps can be sized and can be transparent or opaque.

All image files used are stored in the keyboard file. You do not need to distribute the image file along with your keyboard.

The TPicture class has no properties or events.

7.3.1.6.1 TPictureMethods

Methods are only available from script.

LoadFromFile

7.3.1.6.1.1 TPicture.LoadFromFile

Loads a new graphics file. The file must be in one of the supported formats (see TPicture).

LoadFromFile(Filename: string)

Description
Call LoadFromFile to load a new graphics file into the TPicture object. The existing graphics, if any, are first cleared. The Filename parameter is a string which must contain the full path and filename for the graphics file to be loaded.

Example
```
Sub LoadNewPicture()
  Kbd.Keyboard.Image1.LoadFromFile( "C:\Program Files\MountFocus\Keyboard\Shift.bmp" )
End Sub
```

© 2002 MountFocus Information Systems

7.3.1.7 TFont

The TFont class holds the type and style of the text for a label object (TmfKBLabel). There are two ways to set the font property in the Keyboard Designer. You can click the ellipsis (...) button in the property editor to bring up a dialog box where you can set the font properties, or you can click the "+" to the left of the font property to expand all the font properties right inside of the property editor.

Important. The font itself is NOT stored in your keyboard files - only the name and style settings for the font. If you deploy your keyboard files on a system that do not support the font you've used, Windows will try to substitute the font with a similar font. It is recommended that you use fonts that are shipped with Windows (like Arial, Courier New or Times New Roman) to avoid this problem.

7.3.1.7.1 TFontProperties

Charset
Color
Height
Name
Pitch
Size
Style

© 2002 MountFocus Information Systems

7.3.1.7.1.1 TFont.Charset

Holds the character set for the font.

Charset: integer

Description
Each typeface (font) in Windows supports one or more character sets. Setting this property will instruct the font to use a particular character set (for example Greek or Russian) when drawing its characters. Note that most fonts only support a few character sets. Click the ellipsis (...) button for the The value for the Charset property can be in the range 0..255. You can also use the Windows defined constants listed below.

Constant	Value	Description
ANSI_CHARSET	0	ANSI characters
ARABIC_CHARSET	178	Arabic characters
BALTIC_CHARSET	186	Baltic characters
CHINESEBIG5_CHARSET	136	Traditional Chinese characters
DEFAULT_CHARSET	1	The font is chosen by using the Name and Size properties. Windows will substitute another font if the one specified does not exists.
EASTEUROPE_CHARSET	238	Includes diacritical marks for eastern European countries.
GB2312_CHARSET	134	Simplified Chinese characters
GREEK_CHARSET	161	Greek characters
HANGEUL_CHARSET	129	Korean characters (Wansung)
HEBREW_CHARSET	177	Hebrew characters
JOHAB_CHARSET	130	Korean characters (Johab)
MAC_CHARSET	77	Macintosh characters
OEM_CHARSET	255	Characters dependent on the code page of the operating system.
RUSSIAN_CHARSET	204	Cryllic characters
SHIFTJIS_CHARSET	128	Japanese characters
SYMBOL_CHARSET	2	Symbolic characters
THAI_CHARSET	222	Thai characters
TURKISH_CHARSET	162	Turkish characters

Script values
You can set the value of this property from script by assigning the numeric code representing the character set you want to use, or by assigning one of the Windows defined constants as a string. When you read the property it will return the Windows defined constant as a string, or the numeric value if it does not match one of the constants.

Example

```
Sub SetGreek( Label )
  If Label.Font.Charset <> "GREEK_CHARSET" Then
```

© 2002 MountFocus Information Systems

```
Label.Font.Charset = 161 ' or 'GREEK_CHARSET'
  End If
End Sub
```

7.3.1.7.1.2 TFont.Color

Sets the color of the font.

Color: -$7FFFFFFF-1..$7FFFFFFF

Description
The font color defines the color of the font text. It can be set to any of the standard colors as well as any numeric color value.

See TmfKBSurface.Color for details on Color values.

7.3.1.7.1.3 TFont.Height

Specifies the height of the font in pixels.

Height: integer

Description
When dealing with fonts, the size is usually specified in points. This property allows you to read and write the font size based on pixels instead of points. Changing this property will automatically recalculate the Size property and vice versa. The height is calculated using this formula:

```
Height = -Size * PixelsPerInch / 72
```

This means that when the Height property is positive, the Size property is negative and when the Height property is negative, the Size property is positive.

Example
```
Function GetFontHeight( Font )
  ' Return the font height in pixels, as a positive value
  GetFontHeight = Abs( Font.Height )
End Function
```

7.3.1.7.1.4 TFont.Name

Holds the name of the typeface of the font.

Name: string

Description
The Name property holds the name of the typeface that the font will use. This name must match the name of a font installed on the system. Note that if the combination of a typeface (font name) and the Style setting (for example bold) is unavailable, Windows will substitute a different font.

Example
```
Sub ChangeFont( Label )
  If Label.Font.Name = "Courier" Then
    Label.Font.Name = "Arial"
  End If
End Sub
```

© 2002 MountFocus Information Systems

7.3.1.7.1.5 TFont.Pitch

Specifies if the characters in the font all have the same width.

Pitch: TFontPitch

TFontPitch = (fpDefault, fpVariable, fpFixed)

Description
Most fonts (like this one) use a variable pitch, which means that each character only uses as much horizontal space (width) as it needs. Fixed width fonts (`like this one`) uses the same amount of horizontal space for every character.

TFontPitch value	Description
fpDefault	Allow the font specified in the Name property to decide if the font is variable or fixed.
fpVariable	The font will be displayed using different width for different characters.
fpFixed	All characters will be displayed using the same width.

Note that setting a variable-pitch font to fpFixed, or a fixed-pitch font to fpVariable will cause Windows to substitue the font for another that it considers to be the best match for all the font characteristics. In some cases this means that changing this property alone has no visual effect.

Script values
This property is treated as a string when used from script. Reading it will return either "fpDefault", "fpVariable" or "fpFixed" and only the same strings are accepted when writing to it.

7.3.1.7.1.6 TFont.Size

Specifies the height of the font in points.

Size: integer

Description
Use the Size property to specify the height of the font in points. When the value is positive Size is interpreted as the size of the characters, and when it is negative size is interpreted as the size of the characters and the internal leading (white space above each line).

The Height property holds the size of the font in pixels. Changing one will automatically recalculate the other. Size is calculated using this formula:

```
Size = Height * 72 / PixelsPerInch
```

This means that when the Height property is positive, the Size property is negative and when the Height property is negative, the Size property is positive.

© 2002 MountFocus Information Systems

7.3.1.7.1.7 TFont.Style

Holds a set of effects to apply to the font.

Style: set of TFontStyle

TFontStyle = (fsBold, fsItalic, fsUnderline, fsStrikeOut)

Description
The Style property is a set which means that it can hold zero or more of the values defined in TFontStyle. In the property editor you can include or exclude each of these values by expanding the Style property (click the + to the left) and setting the "property" for each value to True (include) or False (exclude).

Script values
Since the Style property is a set, reading and writing this property from script is slightly different than for other properties. When you read the property, the string returned will contain all styles that are set for the font separated by a comma (,). For example, a style that has its bold and underline attributes set will return "fsBold,fsUnderline". In the same manner you can write to the property by assigning a string containing the styles you want to apply, separated by comma.

Finally you can add a "+" or "-" in front of each style to force that style to be added or removed. This has the added benefit of letting you change one of the attributes without worrying about the others.

Example
```
Sub KeyboardLoad( Sender )
  ' Initialize the Label1 style to italic
  Kbd.Keyboard.Label1.Font.Style = "fsItalic"
End Sub

Sub Label1ButtonDown( Sender )
  ' Make the text bold while the mouse button is down
  ' but leave any other styles alone - i.e. just add bold
  Sender.Font.Style = "+fsBold"
End Sub

Sub Label1ButtonUp( Sender )
  ' Remove the bold attribute, but keep any others
  Sender.Font.Style = "-fsBold"
End Sub
```

© 2002 MountFocus Information Systems