ECL System Tray Icon Contents

ECLTray Overview

See Also

ECLTray System Tray Icon Notification is an ActiveX control written for Visual Basic Projects. The control lets you add icon to the system tray notification area that will respond to mouse and keyboard activity. The control is invisible at run time and serves as a form extension of your project. Note that ECLTray was written using VB5 with Service Pack 3 (VBSP3) installed, the compatibility with the earlier version can not be guaranteed. The service pack can be downloaded from many sites and maybe from my Web pages (under construction as of this writing). Most of the controls with similar features I found on the web do not work with SP3 installed, so I decided to write a control of my own that will work with SP3.

More information can be found by selecting the control and press the About Button on the properties window. Note that this control is release as a freeware, thus you can use this control in any project without any copyright violation. However, sending a copy of your program using this control will be greatly appreciated. If you want to thank me in greater manner, it is completely up to you.

For more information and latest release, see contacting the Author

Related Topic

- Setting the Tray Icon
- Setting the Icon ToolTip Text
- Showing/Hiding the Icon
- MouseDown, MouseUp Event
- MouseMove Event
- DblClick Event
- Contacting the Author

Adding ECLTray Control on a Project

See Also

Before you can use the control, it must be properly registered with the system. You can do this by copying the control to the Windows System directory (Usually 'C:\Windows\System'). Actualy, the control can be copied to anywhere in your drive but the system directory is the proper place to copy it. From a Dos Box, type Regsvr32 {location}\ECLTray.ocx, a dialog box should appear confirming the registration state of the control. To remove the control from the system, add '/u' to the command and delete the control from the system.

To add the control to a new project or an existing project, start Visual Basic and select new or open existing project. From the Project pull down menu, select the components command or press Ctrl+T to display the components window. Select 'ECL System Tray Icon Control' from the list. If the list does not include this control, select browse and locate the ECLtray.ocx. Press the OK button to close the components window. The ECLIPSE Icon should appear in your toolbox confirming the addition of the control. Select the control from the toolbox and place it on your form. Once the control is placed on your form, the About box will appear, displaying some information. Note that this about box will only appear when you add a new control on your form and will not appear during run time, unless you invoke it.

The Control is invisible at run time and can not be resized. The control also has default Icon, the VB5 Icon, so if you don't specify an icon, the default will be shown in the tray. See Setting the Tray Icon for more information.

Setting the Tray Icon

See Also

Returns or sets the Icon to be display on the System Tray Notification area.

Syntax

Set ECLTray.**TrayIcon**

Remarks

The Image that should be assigned to the control must be an Icon type. This is a Windows limitation. Any standard or custom size icon can be assigned to the control but the control will resize the Icon to 16x16 pixels by 16 colors (another Windows limitation). The control has the VB Icon as the default icon. The change or clear this icon use any of the following syntax in your code:

```
Set ECLTray.TrayIcon = Form1.Icon                    'query and copy the icon of the form1
Set ECLTray.TrayIcon = LoadPicture(MyIcon.ico)       'loads an Icon from file
Set ECLTray.TrayIcon = LoadPicture()                 'clear/erase the Icon
Set Form1.Icon = ECLTray.TrayIcon                    'query the control's icon
```

To set the icon during design time, go to the property page and select the control. Choose the TrayIcon Property. A dialog box will appear prompting you for the file that contains the image. Note that this file must be an Icon Type (file with an ICO extension).

Note You can not see the Tray Icon during design time. To see the icon, run the project by pressing the F5 key or choosing the run button from VB toolbar.

Setting the Icon ToolTip Text

See Also

Returns or sets the text to be display when the you pause the mouse pointer over the icon on the System Tray Notification area.

Syntax

ECLTray.**IconToolTip** [=s*tring*]

The ToolTipText property syntax has these parts:

Part	Description
String	A string associated with the control that appears in a small rectangle above the icon when the user's cursor hovers over the object at run time for about one second

At design time you can set the IconToolTip property string in the control's **properties dialog box.**

The change or clear this text, use any of the following syntax in your code:

```
ECLTray.IconToolTip = “My ToolTip”        ‘sets a new IconToolTip text
ECLTray.IconToolTip = “”                  ‘clear the IconToolTip text
```

Note You can not see the Tray Icon during design time. To see the icon, run the project by pressing the F5 key or choosing the run button from VB toolbar.

Showing/Hiding Icon

See Also

Returns or sets a value that determines whether the Icon appears in the Windows 95 taskbar Notification Area.

Syntax

ECLTray.**ShowIcon**

Settings

The settings for the ShowIcon property are:

Setting	Description
True	(Default) The Icon appears in the taskBar notification area
False	The Icon doesn't appear in the taskbar

Note You can not see the Tray Icon during design time. To see the icon, run the project by pressing the F5 key or choosing the run button from VB toolbar.

MouseDown, MouseUp Event

See Also

Occur when the user presses (MouseDown) or releases (MouseUp) a mouse button.

Syntax

Private Sub ECLTray_**MouseDown**([index button As Integer,] shift As Integer, x As Single, y As Single)

Private Sub ECLTray_**MouseUp**([index button As Integer,] shift As Integer, x As Single, y As Single)

The MouseDown and MouseUp event syntaxes have these parts:

Part	Description
Index	Identifies the control if it's in a control array.
Button	Returns an integer that identifies the button that was pressed (MouseDown) or released (MouseUp) to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x, y	Returns a number that specifies the current location of the mouse pointer. The x and y values are always expressed in terms of the screen coordinate.

Remarks

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a given mouse button is pressed or released. Unlike the DblClick events, MouseDown and MouseUp events enable you to distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

The following applies to both DblClick events:

- If a mouse button is pressed while the pointer is over a form or control, that object "captures" the mouse and receives all mouse events up to and including the last MouseUp event. This implies that the x, y mouse-pointer coordinates returned by a mouse event may not always be in the internal area of the object that receives them.
- If mouse buttons are pressed in succession, the object that captures the mouse after the first press receives all mouse events until all buttons are released.

Note You can use a MouseMove event procedure to respond to an event caused by moving the mouse. The button argument for MouseDown and MouseUp differs from the button argument used for MouseMove. For MouseDown and MouseUp, the button argument indicates exactly one button per

event, whereas for MouseMove, the bit is set to 0.

MouseMove Event

See Also

Occurs when the user moves the mouse over the icon.

Syntax

Private Sub ECLTray_**MouseMove**([index button As Integer,] shift As Integer, x As Single, y As Single)

The MouseMove event syntax has these parts:

Part	Description
Index	Identifies the control if it's in a control array.
Button	Returns an integer that identifies the button that was pressed (MouseDown) or released (MouseUp) to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	Returns an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x, y	Returns a number that specifies the current location of the mouse pointer. The x and y values are always expressed in terms of the screen coordinate.

Remarks

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders.

The button argument for MouseMove differs from the button argument for MouseDown and MouseUp. For MouseMove, the button argument is set to 0; button states are ignored. For MouseDown and MouseUp, the button argument indicates exactly one button per event.

Any time you move a window inside a MouseMove event, it can cause a cascading event. MouseMove events are generated when the window moves underneath the pointer. A MouseMove event can be generated even if the mouse is perfectly stationary.

DblClick Event

See Also

Occurs when the user presses and releases a mouse button and then presses and releases it again over an object.

The DblClick event occurs when the user:

- Double-clicks a control with the left mouse button.

Syntax

Private Sub ECLTray_**DblClick** ([index As Integer], button as integer)

Part	Description
Index	Identifies the control if it's in a control array.
Button	Returns an integer that identifies the button that was pressed (MouseDown) or released (MouseUp) to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.

Obtaining the Key State

The state of the Shift, Control, and Alt keys during the DoubleClick event can be determined by retrieving the value of the control property KeyState:

```
X = Ecltray1.KeyState
```

This will return an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.

Remarks

The argument Index uniquely identifies a control if it's in a control array. You can use a DblClick event procedure for an implied action, such as double-clicking an icon to open a window or document.

Note For those objects that receive Mouse events, the events occur in this order: MouseDown, MouseUp, Click, DblClick, and MouseUp.

If DblClick doesn't occur within the system's double-click time limit, the object recognizes another Click event. The double-click time limit may vary because the user can set the double-click speed in the Control Panel. When you're attaching procedures for these related events, be sure that their actions don't conflict. Controls that don't receive DblClick events may receive two clicks instead of a DblClick.

Contacting the Author

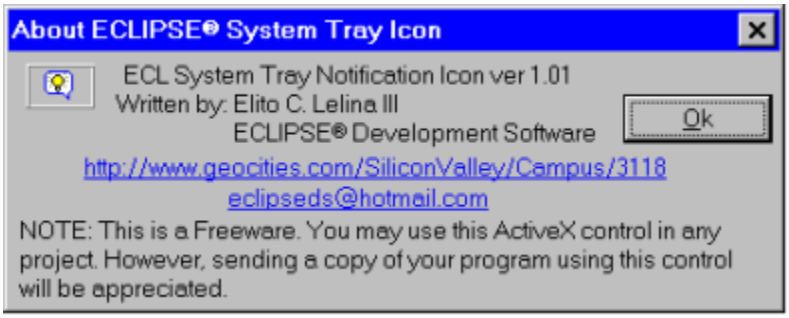
See Also

To contact the author, it is best to select the About property of the control as this will be updated more often than this help file. My web page is currently under construction as of this writing, updates and latest release will be posted to this page.

The About box will always contain a link to contact the author. Please select this link to report bugs, comments, suggestions, appreciation ……. etc. Thanking me in greater manner is completely up to you.

The About box will appear every time you put a new control to a form. This will only happen during design time and not during run time.

The Current About box is shown below.



Related Topic

- Setting the Icon ToolTip Text
- Showing/Hiding the Icon
- MouseDown, MouseUp Event
- MouseMove Event
- DblClick Event
- Contacting the Author

Related Topic

- Setting the Tray Icon
- Showing/Hiding the Icon
- MouseDown, MouseUp Event
- MouseMove Event
- DblClick Event
- Contacting the Author

Related Topic

- Setting the Tray Icon
- Setting the Icon ToolTip Text
- MouseDown, MouseUp Event
- MouseMove Event
- DblClick Event
- Contacting the Author

Related Topic

- Setting the Tray Icon
- Setting the Icon ToolTip Text
- Showing/Hiding the Icon
- MouseMove Event
- DblClick Event
- Contacting the Author

Related Topic

- Setting the Tray Icon
- Setting the Icon ToolTip Text
- Showing/Hiding the Icon
- MouseDown, MouseUp Event
- DblClick Event
- Contacting the Author

Related Topic

- Setting the Tray Icon
- Setting the Icon ToolTip Text
- Showing/Hiding the Icon
- MouseDown, MouseUp Event
- MouseMove Event
- Contacting the Author

Related Topic

- Setting the Tray Icon
- Setting the Icon ToolTip Text
- Showing/Hiding the Icon
- MouseDown, MouseUp Event
- MouseMove Event
- DblClick Event