Charon Software SysTray Control

Please visit the SysTray Control home page on the Internet at
http://www.charonsoftware.com/activex/systray/

Product Description and Features

The SysTray Control is a 32-bit **ActiveX control** for Visual Basic (or any other ActiveX host) that allows you to ***safely* place one or more icons in the system tray** (bottom-right corner of the screen). You can...

- **Change tooltips and icons on the fly,**
- **Safely pop-up menus from the system tray,**
- **Respond to click and double-click events from all three mouse buttons,**
- **Minimize windows to and from the system tray,**
- **Animate your icon with a single line of code,**
- **or even place multiple icons in the system tray at once!**

Sometimes, you create programs that you want to run in the background or unattended, but you still need a way for the user to access your program. Implementing a system tray icon is perfect for this type of application; and that is just one example of the **power and flexibility** you instantly gain with the SysTray control.

Prices and Ordering (with Instant Delivery)

You will receive your registration code **immediately** via e-mail.

To instantly order the SysTray Control online, visit:
http://www.charonsoftware.com/order/systray/

. . . or click here to find out how to order by phone, mail, etc.
($49.00 per developer, all upgrades are free)

Licensing Information

You may distribute this control in your compiled applications only if you have purchased the registered version. One license allows this control to be used by one developer and one developer only at any given time. You may, however, install this control on a network or on multiple computers as long as you have purchased one license for each person using it in a development environment at the same time.

SysTray is licensed for end-user applications only. If you use SysTray as a component in your own controls, there must be a valid SysTray license installed on each development machine using them.

There are no runtime royalties for using this control! If you have purchased the registered version, you may include the OCX with all your applications without any royalty fees to pay! Also, upgrades to this control are always free. To check for an upgrade to this control, please visit http://www.charonsoftware.com/activex/systray/

Ordering the SysTray Control

Thank you for purchasing the Charon Software SysTray Control. You may order online or by telephone, mail, or fax.

For instructions on registering the SysTray control after your order, please click here.

SysTray Control v1.2

The SysTray control is licensed on a per-developer basis. You may use as many copies of the control as you wish but you must purchase one copy of the SysTray Control for each developer using it.

Cost: $49.00 per developer; all upgrades are free.

To instantly order the SysTray Control online, please visit:
http://www.charonsoftware.com/order/systray/

To order by Phone, call NorthStar Solutions
Calls are taken from 9 a.m. to 7 p.m., CST, Monday-Friday
(800)699-6395 (from the U.S. only)
(785)539-3731 (local/international)
Refer to product # **2810**

To order by FAX: (available 24 hours a day)
(785)539-3743
Refer to product # **2810**

To order by E-mail:
Internet: **starmail@nstarsolutions.com**
CompuServe: **starmail**
America Online: **starmail**
Refer to product # **2810**

To order by Regular Mail:
You may pay with a check or money order.
Please make it payable to "**NorthStar Solutions**" and send it to:
1228 Westloop, Suite 204
Manhattan, KS 66502
Refer to product # **2810**

Notes on Registration

How to Enter Your Registration Code

Once you order the SysTray control, you will receive a registration code. To enter your registration code, follow these simple steps:

1. In the Visual Basic environment, click on any SysTray control.
2. Go to your Properties box and select **About**.
3. In the about information box, there is an "Enter Registration Code..." button; press this button.
4. Enter the person or company to whom this control is licensed and your registration code.
5. Press the OK button.

At this point, the SysTray control will acknowledge that you have successfully registered it.

When you add a SysTray control to subsequent projects and/or forms, it will automatically know and remember your registration information.

Special Notes for Users Registering SysTray

If you have been using the SysTray control unregistered in various projects, you will need to make sure that each control is registered properly. The SysTray control stores your registration information inside itself and, when you compile your project, inside your executable. To make sure that each of your SysTray controls are registered, please follow the following steps for each of your SysTray controls. If you do not, some of your SysTray controls may remain unregistered until you do so.

1. Click on the SysTray control.
2. Go to the Properties box and select **About**.
3. (the SysTray control will reload its registration information)
4. The registration information shown here should now be correct.
5. Press the Close button.

When you see that each control is registered correctly, save your project. This will ensure that your controls remember they are registered.

AfterMinimize Event

This event occurs when a user clicks the minimize button on a form that was being watched (see also the **AddMinimizeWatch** method). This event occurs after the window "zooms" to the system tray.

Syntax

Private Sub csSysTray_AfterMinimize(*hWnd* **As Long)**
Private Sub csSysTray_AfterMinimize([*index* **As Integer,]** *hWnd* **As Long)**

The **AfterMinimize event syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with.
index	An integer that uniquely identifies a control if it's in a control array.
hWnd	A handle to the form that was being minimized.

Examples

Example 1 (Responding to an AfterMinimize event with a confirmation message)

```
Private Sub csSysTray1_AfterMinimize (hWnd As Long)
MsgBox "Window was minimized!"
End Sub
```

Remarks

You probably want to set the **ShowInTaskBar property of the forms you are watching to False at design time.**

Also, the Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

Note Because the value of a window handle (hWnd) can change while a program is running, never store the hWnd value in a variable.

Click, MiddleClick, and RightClick Events

This event occurs when the user clicks on the system tray icon with the left, middle, or right mouse button, respectively.

Syntax

Private Sub csSysTray_Click()
Private Sub csSysTray_Click([*index* **As Integer])**
Private Sub csSysTray_MiddleClick()
Private Sub csSysTray_MiddleClick([*index* **As Integer])**
Private Sub csSysTray_RightClick()
Private Sub csSysTray_RightClick([*index* **As Integer])**

The **Click event syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with.
index	An integer that uniquely identifies a control if it's in a control array.

Examples

Example 1 (Responding to a Click event with a confirmation message)

```
Private Sub csSysTray1_Click ()
MsgBox "SysTray icon was clicked (left mouse button)!"
End Sub
```

DblClick, MiddleDblClick, and RightDblClick Events

This event occurs when the user double-clicks on the system tray icon with the left, middle, or right mouse button, respectively.

Syntax

Private Sub csSysTray_DblClick()
Private Sub csSysTray_DblClick([*index* **As Integer])**
Private Sub csSysTray_MiddleDblClick()
Private Sub csSysTray_MiddleDblClick([*index* **As Integer])**
Private Sub csSysTray_RightDblClick()
Private Sub csSysTray_RightDblClick([*index* **As Integer])**

The **DblClick event syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with.
index	An integer that uniquely identifies a control if it's in a control array.

Examples

Example 1 (Responding to a DblClick event with a confirmation message)

```
Private Sub csSysTray1_DblClick ()
MsgBox "SysTray icon was double-clicked (left mouse button)!"
End Sub
```

MouseDown Event

This event occurs when the user presses a mouse button over the system tray icon.

Syntax

Private Sub csSysTray_MouseDown(button As Integer)
Private Sub csSysTray_MouseDown([*index* As Integer,]button As Integer)

The **MouseDown event syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with.
index	An integer that uniquely identifies a control if it's in a control array.
button	Returns an integer that identifies the button that was pressed 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. Buttons are OR'd together to indicate which button(s) caused the event.

Quick Tip

The numbers that represent the button values (1, 2, and 4) are built-in Visual Basic constants. You can use **vbLeftButton instead of 1, vbRightButton instead of 2, and vbMiddleButton instead of 4.**

Examples

Example 1 (Responding to a MouseDown event with a message for each button pressed)

```
Private Sub csSysTray1_MouseDown (Button As Integer)

If (Button And vbLeftButton) Then 'vbLeftButton = 1
MsgBox "Left Button Pressed"
End If
If (Button And vbRightButton) Then 'vbRightButton = 2
MsgBox "Right Button Pressed"
End If
If (Button And vbMiddleButton) Then 'vbMiddleButton = 4
MsgBox "Middle Button Pressed"
End If

End Sub
```

MouseMove Event

This event occurs when the user moves the mouse cursor over the icon in the system tray.

Syntax

Private Sub csSysTray_MouseMove()
Private Sub csSysTray_MouseMove([*index* **As Integer])**

The **MouseMove event syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with.
index	An integer that uniquely identifies a control if it's in a control array.

Examples

Example 1 (Responding to a MouseMove event with a confirmation message)

```
Private Sub csSysTray1_MouseMove ()
MsgBox "Mouse Cursor Moved Over SysTray Icon!"
End Sub
```

MouseUp Event

This event occurs when the user releases a mouse button over the system tray icon.

Syntax

Private Sub csSysTray_MouseUp(button As Integer)
Private Sub csSysTray_MouseUp([*index* As Integer,]button As Integer)

The **MouseUp event syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with.
index	An integer that uniquely identifies a control if it's in a control array.
button	Returns an integer that identifies the button that was released 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. Buttons are OR'd together to indicate which button(s) caused the event.

Quick Tip

The numbers that represent the button values (1, 2, and 4) are built-in Visual Basic constants. You can use **vbLeftButton instead of 1, vbRightButton instead of 2, and vbMiddleButton instead of 4.**

Examples

Example 1 (Responding to a MouseUp event with a message for each button released)

```
Private Sub csSysTray1_MouseUp (Button As Integer)

If (Button And vbLeftButton) Then 'vbLeftButton = 1
MsgBox "Left Button Released"
End If
If (Button And vbRightButton) Then 'vbRightButton = 2
MsgBox "Right Button Released"
End If
If (Button And vbMiddleButton) Then 'vbMiddleButton = 4
MsgBox "Middle Button Released"
End If

End Sub
```

AboutBox Method

Displays the About box, including registration information, for the specific SysTray control.

Syntax

csSysTray[(index)].**AboutBox**

Examples

Example 1 (showing the SysTray about box at run time)

csSysTray1.AboutBox

Remarks

This is the same as clicking About in the Properties window except that you usually cannot edit the registration information for the control.

PopupMenu Method

Displays a pop-up menu at the current mouse location or at specified coordinates. Doesn't support named arguments.

Special Note: If you use the regular Visual Basic PopupMenu method, your system tray popup menus will probably not go away properly when the user clicks somewhere else on the screen. Whenever you want to popup a menu from the system tray, it is strongly advised you use the PopupMenu method of the SysTray control instead.

Syntax

csSysTray[(index)].**PopupMenu MenuName, [Flags], [X], [Y], [DefaultMenu]**

The **PopupMenu method syntax has these parts:**

Part	Description
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
MenuName	Required. The name of the pop-up menu to be displayed. The specified menu must have at least one submenu (item on it).
Flags	Optional. A value or constant that specifies the location and behavior of a pop-up menu, as described in Settings.
X	Optional. Specifies the x-coordinate where the pop-up menu is displayed. If omitted, the mouse coordinate is used.
Y	Optional. Specifies the y-coordinate where the pop-up menu is displayed. If omitted, the mouse coordinate is used.
DefaultMenu	Optional. Specifies the name of a menu control in the pop-up menu to display its caption in bold text. If omitted, no controls in the pop-up menu appear in bold. Making an item in a menu bold usually signifies to the user that it is the defualt item (they would get by left-clicking on the system tray icon).

Settings

The settings for *Flags are:*

Constant (location)	Value	Description
vbPopupMenuLeftAlign	0	(Default) The left side of the pop-up menu is located at x.
vbPopupMenuCenterAlign	4	The pop-up menu is centered at x.
vbPopupMenuRightAlign	8	The right side of the pop-up menu is located at x.

Constant (behavior)	Value	Description
vbPopupMenuLeftButton	0	(Default) An item on the pop-up menu reacts to a mouse click only when button.
vbPopupMenuRightButton	2	(Recommended) An item on the pop-up menu reacts to a mouse click wi or the left mouse button.

To specify two flags, combine one constant from each group using the Or operator.

Examples

Example 1 (Popping up a menu from a system tray icon when it is right-clicked)

```
Private Sub csSysTray1_RightClick()

'we pass in vbPopupMenuRightButon as a flag so our popup menu works with both buttons.
csSysTray1.PopupMenu mnuSysTray, vbPopupMenuRightButton

End Sub
```

Example 2 (Popping up a menu again, but this time setting mnuHelpAbout to be bold, the default)

```
Private Sub csSysTray1_RightClick()

'we pass in vbPopupMenuRightButon as a flag so our popup menu works with both buttons.
csSysTray1.PopupMenu mnuSysTray, vbPopupMenuRightButton, , , mnuHelpAbout

End Sub
```

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

You specify the unit of measure for the x and y coordinates using the **ScaleMode property. The x and y coordinates define where the pop-up is displayed relative to the specified form. If the x and y coordinates aren't included, the pop-up menu is displayed at the current location of the mouse pointer.**

When you display a pop-up menu, the code following the call to the **PopupMenu method isn't executed until the user either chooses a command from the menu (in which case the code for that command's Click event is executed before the code following the PopupMenu statement) or cancels the menu. In addition, only one pop-up menu can be displayed at a time; therefore, calls to this method are ignored if a pop-up menu is already displayed or if a pull-down menu is open.**

RemoveMinimizeWatch Method

This method disables minimize "watching" of the window (form) passed in. When the user presses the minimize button on the "watched" form, the window will no longer zoom to the system tray instead.

Syntax

csSysTray[(index)].**RemoveMinimizeWatch hWnd**

The **RemoveMinimizeWatch method syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
hWnd	A handle to the form you want to stop being watched. (ex: Form1.hWnd)

Examples

Example 1 (Ending a Minimize Watch on a form)

csSysTray1.RemoveMinimizeWatch Form1.hWnd

Remarks

You probably want to set the **ShowInTaskBar property of the forms you are watching to False at design time. See also the AddMinimizeWatch method.**

Also, the Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the hWnd value in a variable.

StopAnimation Method

This method stops animating the system tray icon.

Syntax

csSysTray[(index)].**StopAnimation**

The **StopAnimation method syntax has these parts:**

Part	Description
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.

Examples

Example 1 (Stopping an Animation)

csSysTray1.StopAnimation

Remarks

To start animating a system tray icon, call the **StartAnimation** method.

AddMinimizeWatch Method

This method enables minimize "watching" of the window (form) passed in. When the user presses the minimize button on the "watched" form, the window will zoom to the system tray instead!

Syntax

csSysTray[(index)].**AddMinimizeWatch hWnd**

The **AddMinimizeWatch method syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
hWnd	A handle to the form you want to be watched. (ex: Form1.hWnd)

Examples

Example 1 (Starting a Minimize Watch on a form)

csSysTray1.AddMinimizeWatch Form1.hWnd

Remarks

You probably want to set the **ShowInTaskBar property of the forms you are watching to False at design time. See also the RemoveMinimizeWatch method.**

Also, the Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the hWnd value in a variable.

StartAnimation Method

This method starts animating the system tray icon using images from the supplied imagelist.

Syntax

csSysTray[(index)].**StartAnimation imgList, [Interval], [StartFrame]**

The **StartAnimation method syntax has these parts:**

Part	Description
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
imgList	Required. An imagelist containing icons to use in animating the system tray icon.
Interval	Optional. The number of milliseconds to show each frame of the attached imagelist before proceeding to the next frame. This is also the amount of time paused between each frame. **Default = 250 (250 milliseconds, 1/4 second)**
StartFrame	Optional. The frame in the imagelist where you wish to start the animation. By default, the animation starts with the first frame.

Examples

Example 1 (Starting an Animation using Defaults)

csSysTray1.StartAnimation imglstDefault

Example 2 (Starting an Animation on frame 2)

csSysTray1.StartAnimation imglstDefault, ,2

Example 3 (Starting an Animation at double speed)

csSysTray1.StartAnimation imglstDefault, 125

Remarks

To stop animating a system tray icon, call the **StopAnimation** method.

TrayHide and TrayShow Methods

This method will hide (or show, respectively) the system tray icon if it is currently visible (or hidden, respectively). See also the **TrayVisible** property.

Syntax

csSysTray[(index)].**TrayHide**
csSysTray[(index)].**TrayShow**

The **TrayHide and TrayShow methods syntax have these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.

Examples

Example 1 (Hiding a System Tray Icon)

```
csSysTray1.TrayHide
```

Example 2 (Showing a System Tray Icon)

```
csSysTray1.TrayShow
```

ZoomFromTray and ZoomToTray Methods

This method causes the window referred to by hWnd to be zoomed (restored or minimized respectively) to or from the system tray.

Note: before calling the ZoomFromTray method, you need to both hide the designated window and place it where you want it to be after zooming.

Syntax

csSysTray[(index)].**ZoomFromTray hWnd**
csSysTray[(index)]**.ZoomToTray** hWnd

The **ZoomFromTray method syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
hWnd	A handle to the form you want zoomed. (ex: Form1.hWnd)

Examples

Example 1 (Zooming a form from the System Tray)

```
Form1.Hide 'make sure the form is hidden
Form1.Move 100,100 'move it where you want it to end up
csSysTray1.ZoomFromTray Form1.hWnd 'call the "zooming" animation.
```

Remarks

You probably want to set the **ShowInTaskBar** property of the forms you are watching to False at design time.

Also, the Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or hWnd. The hWnd property is used with Windows API calls. Many Windows operating environment functions require the hWnd of the active window as an argument.

Note Because the value of this property can change while a program is running, never store the hWnd value in a variable.

TrayIcon Property

This property contains the icon shown in the system tray. This icon is also shown at design time in the right half of the SysTray control. This must be an icon (.ico or .cur file).

Syntax

Set csSysTray[(index)].TrayIcon = icon

The **TrayIcon property syntax has these parts:**

Part	**Description**
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
icon	An icon containing the graphic you wish to use in the system tray.

Settings

The settings for *icon are:*

Setting	**Description**
(None)	(Default) No icon.
(.ico or .cur file)	Specifies an icon. You can load the icon from the Properties window at design time. At run time, you can also set this property using the **LoadPicture function on an icon.**

Examples

Example 1 (setting the system tray icon at run time)

Set csSysTray1.TrayIcon=pctMyPicture.Picture

Example 2 (retrieving the system tray icon at run time)

Set pctMyPicture.Picture=csSysTray1.TrayIcon

Remarks

When setting the **TrayIcon p**roperty at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic isn't saved with the application. Use the **SavePicture** statement to save a graphic from a SysTray control into a file.

Note At run time, the **TrayIcon c**an be set to any other object's **DragIcon** or **Icon** property, or you can assign it the graphic returned by **LoadPicture.**

TrayTip Property

This property contains the tooltip shown by the system tray icon when a user holds the mouse cursor over it.

Syntax

csSysTray[(index)].**TrayTip [= string]**

The **TrayTip property syntax has these parts:**

Part	Description
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
string	A string associated with the SysTray control that appears in a small rectangle above the system tray icon when the user's cursor hovers over it at run time for about one second.

Examples

Example 1 (setting the system tray tooltip (traytip) at run time)

csSysTray1.TrayTip="This is a system tray tooltip!"

Example 2 (retrieving the system tray tooltip (traytip) at run time)

sTemp = csSysTray1.TrayTip

Remarks

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

At run time, you may change the system tray tooltip (traytip) on the fly, at any time.

TrayVisible Property

Sets or returns whether the icon is visible in the system tray. See also the **TrayShow and TrayHide methods.**

Syntax

csSysTray[(index)].**TrayVisible [= boolean]**

The **TrayVisible property syntax has these parts:**

Part	Description
csSysTray	The SysTray control you are working with, or a control array of SysTray controls.
boolean	A Boolean expression specifying whether the system tray icon is visible or hidden.

Settings

The settings for *boolean are:*

Setting	Description
True	(Default) The system tray icon is visible (when the form that contains it is loaded).
False	The system tray icon is hidden.

Examples

Example 1 (showing/hiding the system tray icon at run time)

csSysTray1.TrayVisible = True

Example 2 (retrieving the visible state of the system tray icon at run time)

trayiconVisible = csSysTray1.TrayVisible

Remarks

To hide a system tray icon at startup, set the TrayVisible property to False at design time. Setting this property in code enables you to hide and later redisplay a system tray icon at run time in response to a particular event.

Note Using the TrayShow or TrayHide method on a SysTray control is the same as setting the control's TrayVisible property in code to True or False, respectively.