Visual Basic Tips & Tricks

VB for DOS

VB for Windows

Help Files

Release History

About Developer

About VB Tips & Tricks

Visual Basic For DOS

Files & Directories

Files & Directories

Finding Directories

Finding Directories

How do you test to see a directory exists with Visual Basic? You might want to use the DIR$ command like this:

```
DIR$("c:\test\*.*)
```

This will work as long as there are files in the directory. But how about if the directory is empty? The above code won't work.

To get around this I use the "nul" specification. Every directory has a "nul" file in it, regardless if there are any other files in it or not. Below is how to use it:

```
XY$ = DIR$("c:\test\nul")
IF XY$ <> "nul" THEN
        MKDIR "c:\test"
END IF
```

Submitted By: David McCarter

Visual Basic For Windows


Buttons & Image Control


Controls


Form_Load () Events


Graphics


Hot Spots


List Boxes


Menus


Miscellaneous


Text Boxes


Tool Boxes


Windows

Buttons & Image Control

Image Control As A Button

Mouse Button Up Or Down Status

Image Control As A Button

One of the easies techniques for adding graphical effects to your program is to use an image control as a button. When the image control receives a Click event, you simply substitute the value of the Picture property. The key to this technique is to define a pair of invisible image controls with pictures corresponding to the up and down statues of the control.

For example, you could create a button that visually represents a locked and unlocked state. One adtantage of using icon files rather than bitmap files is that any underlying image shows trough the mask area of the icon .

When the for is loaded, the Form_Load event procedure sets the appropriate image in the image control:

```
Sub Form_Load()
        Padlock.Picture = LockOpen.Picture
End Sub
```

The image control responds to the click event by replacing the picture in the control:

```
Sub Padlock_Click()
Static LockedFlag As Integer
If LockedFlag Then
        Padlock.Picture = LockOpen.Picture
        Else
                Padlock.Picture = LockClosed.Picture
End If
LockedFlag = Not LockedFlag
End Sub
```

Source: Microsoft Developer Network News, July 1993

Mouse Button Up Or Down Status

Here is how to check the mouse button 'up_or_down' status.

Declare Function GetKeyState Lib "user" (ByVal k%) As Integer
Declare Function GetAsyncKeyState Lib "user" (ByVal k%) As Integer

This functions will tell you whether a key is up or down, and if it's "toggled". The high order bit tells you if the key is up or down. The low order bit tells you if it's toggled - each time the key is pressed and then released the low order bit will change.

The first function maintains the keyboard state when the last *window message* was received by the app and is the one you normally use. The second one corresponds to the real time keyboard state.

Now, why on earth do I talk keyboard when the question was about the mouse? - Thats because the mouse buttons are considered as "virtual" keyboard keys. Some virtual key codes are:

VK_LBUTTON 1
VK_RBUTTON 2
VK_MBUTTON 4
VK_TAB 9
VK_ESCAPE 27

So to wait for the left mouse button to be first pressed and then released you could use:

Do While GetKeyState(VK_LBUTTON)>=0: DoEvents: Loop
Do While GetKeyState(VK_LBUTTON)<0: DoEvents: Loop

Submitted By: Dan Bystrom - InterNet:adbbyd@msmail.hk-r.se

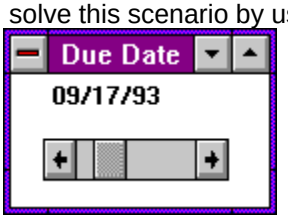
Controls

Using Controls Indirectly To Change Data

Using Controls Indirectly To Change Data

In certain cases, you might not want the user to change data in a control directly, but *indirectly* using another control. this technique can minimize, or even eliminate the process of validating user input.

Let's say you want the user to enter a due date that is from 1 to 30 days from today's date. You can solve this scenario by using a horizontal scroll bar that modifies the Caption property of a label:



The scroll bar property settings are:

LargeChange=5
Min=1
Max=30

Every time you change the scroll bar value, the HScroll1_Change event procedure is generated, which in turn calls the DisplayDate procedure:

```
Sub HScroll1_ Change ()
        DisplayDate
End Sub

Sub DisplayDate ()
        TimeVal = Now + HScroll1.Value
        Label1.Caption = Format(timeVal, "mm/dd/yy")
End Sub
```

The *TimeVal* variable contains the current time returned from the **Now** function plus the current setting of the scroll bar's Value property (1-30). In Visual Basic, adding integer values to a time value increments the time value by days.

Lastly, to set the initial date value on the label, you need to call the DisplayDate procedure when the form is loaded:

```
Sub Form_Load ()
        DisplayDate
End Sub
```

VB for Windows 3.0 Tested!

Source: Microsoft Developer Network News, September 1993

List Boxes

Tab Stops In A List Box

Tab Stops In A List Box

The standard Visual Basic list box supports tab stops. This means that if the string value you add to the list box contains tab characters, the tabs cause the list box to display columns of strings.

Here's how to add the first item to the list box:

```
t=Chr$(9)
name="Michael Cage"
list1.AddItem name + t + "44" + t + "C/F"
```

You could also narrow the width of the list box so that you don't display the second and third columns. This techique allows you to store multiple strings per item, while only displaying the first string. Notice, however, that you would need to write additional code to extract specific strings.

Taking the idea of a list box as a storage mechanism one step further, you could make the list box invisible and only refer to it in your code.

Source: Microsoft Developer Network News, July 1993

Graphics

Decoding Binary CGM Graphics

Decoding Binary CGM Graphics

To decode binary CGM (Computer Graphics Metafile) files in visual basic 3.0 std. The binary CGM standard specifies byte-order and bit-order for multi-byte binary numbers as left-to-right (big-Endian), but the PC byte order is not left-to-right, it is little-Endian.

Byte-swapping is simple in C or C++, but not so simple in visual basic. (Problems with sign extension in signed integer types, and no unsigned integer types.) Thus, I used the following function to swap two-byte integers:

```
Function SwapBytes (num As Integer) As Integer
' Take an input integer, assumed to be in "left to right" byte order, and convert it to "standard"
Intel format by swapping the two bytes.
Dim TextVal As String
Dim NewTextVal As String
Dim StringLength As Integer

TextVal = Hex$(num)
StringLength = Len(TextVal)
Select Case StringLength
Case 1
        NewTextVal = "&H" & "0" & TextVal & "00"
Case 2
        NewTextVal = "&H" & TextVal & "00"
Case 3
        NewTextVal = "&H" & Right$(TextVal, 2) & "0" & Left$(TextVal, 1)
Case 4
        NewTextVal = "&H" & Right$(TextVal, 2) & Left$(TextVal, 2)
End Select
SwapBytes = Val(NewTextVal)
End Function
```

Submitted By: Steven W. Layten - InterNet: swl26@cas.org

Hot Spots

Menus

Pop-up Menus

Pop-up Menus

One of the new features of Visual Basic 3.0 is pop-up menus. You can easily create a pop-up menu from an existing menu structure. Let's look at how you would create a pop-up menu from the tradional menu in the Blanker sample application provided with Visual Basic (in your VB\SAMPLES\GRAPHICS directory).

First, use Menu Design windows to set the Visible property of mnuOption to False. Then, add the following event procedure to display the pop-up menu:

```
Sub Form_MouseUp (Button As Integer,...)
If Button = 2 Then
        PopupMenu mnuOption
End If
End Sub
```

Notice that the Form_MouseUp event procedure uses the right mouse button to display the menu.

Source: Microsoft Developer Network News, July 1993

Text Boxes

Masking Input

Masking Input

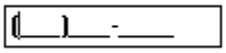
For greater control of data input, you might consider using the masked edit control. This control is included with the Visual Basic Professional Edition, along with many other custom controls.

The masked edit control provides a Mask property that lets you specify an input mask of literals and place holders. Literals provide visual cues about the type of data being used - for example, parentheses indicate a telephone area code. Placeholders represent a specific type of required value - for example, the # symbol indicates that you can only enter a decimal value (0-9).

Here's how you might set the Mask property to indicate a telephone number:

Mask (###)###-####

At run time, the insertion point moves to the first placeholder:



The underscores in the control represent characters to enter. In this case, you can enter only numeric characters.

VB for Windows 3.0 Tested!

Source: Microsoft Developer Network News, September 1993

Tool Boxes

To create a tool box for your application, simply set up a form as a parent and another form as your toolbox/floating dialog whatever. In a suitable declarations section declare the API function as follows:

```
Declare Function SetParent% Lib "User" (ByVal hWndChild%, ByVal hWndNewParent%)
```

For a floating toolbox form name of tbox to make the toolbox form name of tbox float over a form name parent try the following in the routine to show the toolbox:

```
Sub ShowTbox_Click ()
Dim ret As Integer
If doshow = False Then 'toolbox not visible
        ret = SetParent(tbox.hWnd, parent.hWnd) 'this makes the toolbox float
        tbox.Left = 0 'sets position to top left corner of parent
        tbox.Top = 0
        tbox.Show 'makes toolbox visible
        'try tbox.show 1 i.e. modal to see what happens
        doshow = True
        Showtbox.Caption = "&Hide Toolbox"
                Else
                tbox.Hide
                doshow = False
                Showtbox.Caption = "&Show Toolbox"
End If
End Sub
```

A couple of small caveats however. If you try tbox.show 1 i.e. modal you'll find the form will show but you will be unable to do anything with it. Secondly you absolutely *MUST* unload the child form i.e. tbox BEFORE unloading the main form otherwise your program will crash.

Submitted By: Matthew Dexter - InterNet:ch01md@surrey.ac.uk

Editors Note: A working code sample of this can be downloaded via ftp on CICA: pub/pc/win3/programr/vbasic/vbtbox.zip

Form_Load () Events

Detecting Previous Instances Of A Program

Detecting Prevous Instances Of A Program

There are times when you may want to prevent a second instance of your program from running. The App opject provides a PrevInstance property that allows you to determine whether a previous instance of the program is running. Here's how you might write your code:

```
Sub Form_Load ()
If App.PrevInstance Then
        msg$ = App.EXEName & " already running "
        MsgBox msg$, 48
        End
End If
End Sub
```

Notice that the procedure uses the EXEName property of the App object to display the program's name in the Visual Basic message box.

Source: Microsoft Developer Network News, July 1993

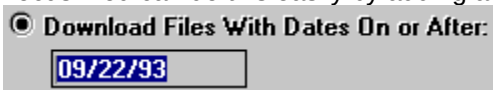
Miscellaneous

 **Automatic Selection Of Text**

 **Easy Help**

Automatic Selection Of Text

When using text boxes, it is often useful to generate automatic selection of text when the control gets focus. You can do this easily by adding a couple of lines to the GotFocus even procedure:




```
Sub Text1_GotFocus ()
Text1.SelStart = 0
Text1.SelLength = 65000
End Sub
```

Notice that the length value for SelLength is 65000, nearly the maxium length allowed in a text bos. This forces Visual Basic to use the actual length of the text as the SelLength.

VB for Windows 3.0 Tested!

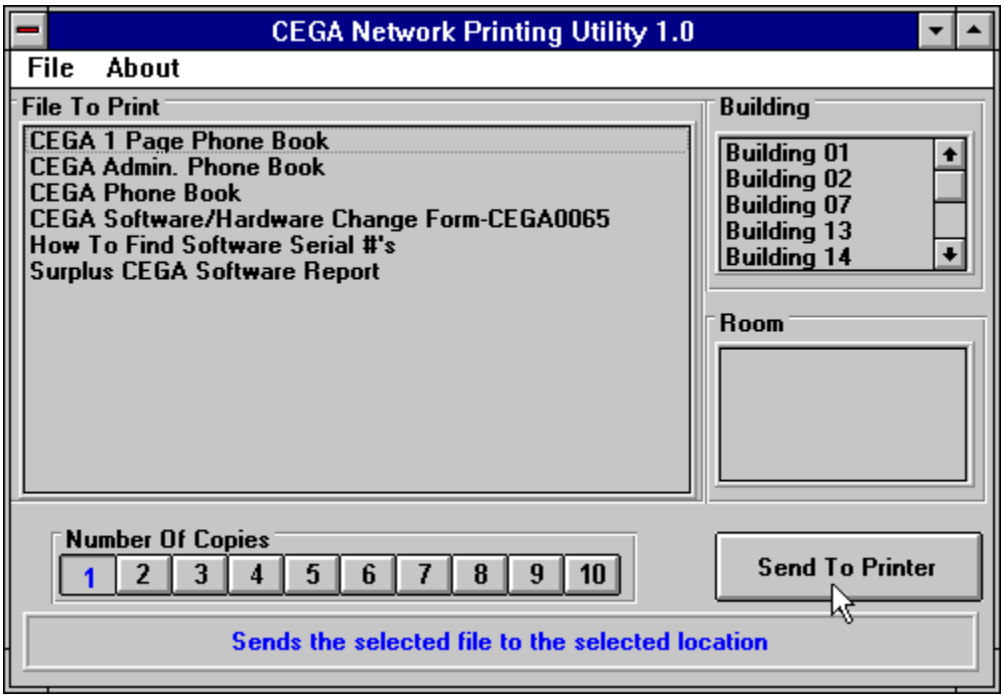
Source Microsoft Developer Network News, July 1993

Easy Help

I created a simple VB for Windows program the print company files to any network printer. I designed it to be easy to operate. When I went to write the help file, I decided against it. This would mean I would have to provide an additional file with the program and the time involved to write the help file was too much. I decided to add a text box with one line of help for each button.

I started by creating a SSPanel using the THREED.VBX file (any type of text box will do just fine). Then using the MouseMove event I added the following lines:

```
Sub Print_Now_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)
HelpText = "Sends the selected file to the selected location"
End Sub
```



It's that simple! You can also use MouseDown, MouseUp,GotFocus events too. I found the MouseMove the easiest to use. Just make sure when you code your program that all the buttons and the form has some kind of help text otherwise the mouse could be over a button and display the wrong help line.

Submitted By: David McCarter

Windows

Keeping A Window On Top

Keeping A Window On Top

To keep a program window on top (always visible) in Visual Basic use a WINAPI function.

Code in Main Module:

```
Declare Sub SetWindowPos Lib "User" (Byval hWnd as integer, Byval hWndInsertAfter as Integer, Byval X as Integer, Byval Y as Integer, Byval cx as Integer, Byval cy as Integer, Byval wFlags as Integer)
```

Code in a Submodule:

```
SetWindowPos form1.hWnd, -1, 0, 0, 0, 0, &H50 'This will make the window always visible!
```

Code in Submodule 2:

```
SetWindowPos form1.hWnd, -2, 0, 0, 0, 0, &H50 'This will put "Always Visible" off!
```

Submitted By: Henk Hakvoort

Windows Help Files

 **Adding Sound To A Help File**

 **Calling WINHELP**

 **Creating Bullets**

Adding Sound To A Help File

To add sound support for .wav files in a windows help file:
In the [CONFIG] section of the project file:

RegisterRoutine("mmsystem","sndPlaySound","Si")

To call the DLL in a macro as a hotspot:

!sndPlaySound("anything.wav",0)

This is a way cool to way to build documents. It *could* be easier though.

Submitted By: Tod Massa - InterNet: MASSATR@SLUVCA.SLU.EDU

Calling WINHELP

Listed below are keywords used by WINHELP and an example on how to use it.

```
'Help engine declarations.
'Commands to pass WinHelp()
Global Const HELP_CONTEXT = &H1 ' Display topic identified by number in Data
Global Const HELP_QUIT = &H2 ' Terminate help
Global Const HELP_INDEX = &H3 ' Display index
Global Const HELP_HELPONHELP = &H4 ' Display help on using help
Global Const HELP_SETINDEX = &H5 ' Set an alternate Index for help file with more than one
index
Global Const HELP_KEY = &H101 ' Display topic for keyword in Data
Global Const HELP_COMMAND = &H102 ' Execute Help macro
Global Const HELP_MULTIKEY = &H201   ' Lookup keyword in alternate table and display topic

Declare Function WinHelp Lib "User" (ByVal hWnd As Integer, ByVal lpHelpFile As String, ByVal
wCommand As Integer, dwData As Any) As Integer

Type MULTIKEYHELP
        mkSize As Integer
        mkKeylist As String * 1
        szKeyphrase As String * 253
End Type

Case 2 ' Help Search Command
        HelpCmd = HELP_COMMAND ' Set help command to bring up search box
        curFile = App.HelpFile
        curData = "Search()" ' Data is macro to be executed by WinHelp

' To make this work, we need to do two help commands in succession, so we'll do one here and
one down below. The commented line 2nd below is the command to be executed to bring up the
search dialog.
result = WinHelp(hWnd, curFile, HELP_INDEX, 0&)
result = WinHelp(hWnd, curFile, HELP_COMMAND, ByVal "Search()")
```

Submitted By: Gary Ferguson - InterNet: GARYFE@MICROSOFT.COM

Creating Bullets

One night I spent at least 2 hours trying to figure out how to create bullet items in a Windows 3.1 Help File. I found some Microsoft documentation which in both places it showed how to do it wrong! Below is the way that I found that works when using the QDHelp shareware program to compile the RTF file:

/para \tx360 \li360 \fi-360
{\f1 \'B7} \tab
Configurable to use many different sizes of diskettes.
Can even use 3 1/2 1.4MB (2.7MB) floppies compressed with Stacker.
/endpara

The '\tx360' RTF command sets the first tab stop to 360. The '\li360' sets the left indent to 360 and the '\fi-360' sets the first line indent to -360 or 0 in this case. The '\f1' sets the font to #1, which should be the Symbol font. The '\'B7' is the hex code for the bullet character in the Symbol font. The '\tab' moves the first line of text to the tab stop created with '\tx360'.

Here is what it looks like:

- Configurable to use many different sizes of diskettes. Can even use 3 1/2 1.4MB (2.7MB) floppies compressed with Stacker.

Submitted By: David McCarter

Release History

Version 1.1 Released on: 10/1/93 - File Name: VBTIPS11.ZIP

Version 1.0 Released on: 9/9/93 - File Name: VBTIPS10.ZIP

Developer Information

The Visual Basic Tips & Tricks Help File

Is compiled by: David McCarter

I can be reached at:

DPM Computer Solutions, 8430-D Summerdale Road, San Diego, CA 92126-5415 USA.

InterNet-MCCART@VAXD.GAT.COM.
Compuserve Users-Contact me using the address: INTERNET:MCCART@VAXD.GAT.COM

If I (or someone that I know) has tested the submitted code to make sure that it works, then we tell you by adding a graphic like this  for the Visual Basic version it was tested with.

Disclaimer: I will try to make sure that all the coding in this help files is correct and works! I am not responsible for any damage that might happen to your computer or programs. REMEMBER...save your work before trying any coding listed here.

This Help File was written with the following Shareware programs:
Quick & Dirty Help
WinEdit

About VB Tips & Tricks

The future of this project rests on you shoulders... I wanted to develop this help file to provide some sort of centralized forum for users to share **VB Tips & Tricks**. I want to pool information from many different sources into one help file.

As you might already know, Microsoft books and help files are (in my opinion) not written very well. They do not contain much help that a normal programmer might want. Some of their information is even printed wrong! So where does one turn? To off-the-shelf books, user forums, magazine articles and the like. Try to keep all that organized to retrieve the information you need quickly. Not so easy, is it?

I want this help file to provide HELP with small **VB Tips & Tricks**. Undocumented ones, work arounds, easier way to do things etc... **But I need your help!**. I need your input!

Please submit your **VB Tips & Tricks** for others to use. Since this help file is freeware or public domain, the only thing you will receive is your name imbedded in this help file forever.

How to submit your VB Tips & Tricks:

Write down your tip or trick explaining exactly what is it is. Provide any coding (make sure it works) and graphics if you want. Text must be is ASCII format and graphics must be 16 color in a BMP format.

Send your submission to:
DPM Computer Solutions
c/o VB Tips & Tricks
8430-D Summerdale Road
San Diego, CA 92126-5415 USA

Please submit them on a 3 1/2 disk using a floppy disk mailer.

You can also save some $$$ and electronically send it to me at:
InterNet - MCCART@VAXD.GAT.COM
CompuServe - INTERNET:MCCART@VAXD.GAT.COM
Send any files (please zip them first to 500K or less) using uuencode or binhex coding.

How To Receive Updates- The easiest way is to receive them electronically through e-mail. To get on the e-mailing list just submit a VB Tip or Trick! Use my e-mail address above and send me your name, e-mail address, coding preference (uuencode or binhex) and file size limitations your system might have along with your VB Tip or Trick. My system has a 500K limit, so if and when the zipped file reaches that size, e-mail will no longer work. This will also be posted on many InterNet systems and BBS systems to be listed in the next version.