Tipp : 1.

IN FORM:

```
'Beispiel : Registrierten Anwender aendern...

Public Sub sCreateNewOwnder(lHKey As Long, sPath As String, sValue As String, sData As String)
   Dim lKey          As Long
   Dim lRet          As Long

   lRet = RegCreateKey(lHKey, sPath, lKey)
   lRet = RegSetValueEx(lKey, sValue, 0, REG_SZ, ByVal sData, Len(sData))
   lRet = RegCloseKey(lKey)
End Sub

Private Sub Command1_Click()
   sOwner$ = Text1
   If sOwner$ = "" Then
          MsgBox "Sie muessen einen zu anderen Namen eingeben", vbOKOnly + vbCritical, "Fehler...", 0, 0
           Exit Sub
   End If
   Call sCreateNewOwnder(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion", "RegisteredOwner",  sOwner$)
End Sub

Private Sub Command2_Click()
   Unload Me
End Sub
```

IN MODUL:

```
Declare Function RegCloseKey Lib "advapi32.dll" (ByVal Hkey As Long) As Long

Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" _
                                   (ByVal Hkey As Long, ByVal lpValueName As String, ByVal _
                                   Reserved As Long, ByVal dwType As Long, lpData As Any, ByVal _
                                   cbData As Long) As Long

Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" _
                                   (ByVal Hkey As Long, ByVal lpSubKey As String, phkResult As Long) As Long

Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const REG_SZ = 1
Public Const REG_DWORD = 4
```

Tipp : 2.

IN FORM:

```
'Beispiel : registrierte Organisation aendern

Public Sub sSaveNewOrganisation(lHKey As Long, sPath As String, sValue As String, sData As String)
   Dim lKey          As Long
   Dim lRet                    As Long

   lRet = RegCreateKey(lHKey, sPath, lKey)
   lRet = RegSetValueEx(lKey, sValue, 0, _
   REG_SZ, ByVal sData, Len(sData))
   lRet = RegCloseKey(lKey)
End Sub

Private Sub Command1_Click()
          sOrganisation$ = Text1
          If sOrganisation$ = "" Then MsgBox "Sie muessen einen neuen Namen eigeben!", _
                                                vbOKOnly + vbCritical, "Fehler...", 0, 0
                Exit Sub
          End If
Call sSaveNewOrganisation(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion", "RegisteredOrganization",
sOrganisation$)
End Sub

Private Sub Command2_Click()
   Unload Me
End Sub
```

IN MODUL:

```
Declare Function RegCloseKey Lib "advapi32.dll" (ByVal Hkey As Long) As Long

Declare Function RegSetValueEx Lib "advapi32.dll" _
                                Alias "RegSetValueExA" _
                                (ByVal Hkey As Long, _
                                ByVal lpValueName As String, _
                                ByVal Reserved As Long, _
                                ByVal dwType As Long, _
                                lpData As Any, _
                                ByVal cbData As Long) _
                                As Long

                                Declare Function RegCreateKey Lib "advapi32.dll" Alias _
                                "RegCreateKeyA" _
                                (ByVal Hkey As Long, _
                                ByVal lpSubKey As String, _
                                phkResult As Long) _
                                As Long

Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const REG_SZ = 1
Public Const REG_DWORD = 4
```

Tipp : 3.

IN FORM:

```
'Beispiel : Programme in der Registrierung speichern
'In diesem Beispiel in:
'sSaveApplicationValues(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion\Run"

Public Sub sSaveApplicationValues(lHKey As Long, sPath As String, sValue As String, sData As String)
   Dim lKey          As Long
   Dim lRet          As Long

   lRet = RegCreateKey(lHKey, sPath, lKey)
   lRet = RegSetValueEx(lKey, sValue, 0, REG_SZ, ByVal sData, Len(sData))
   lRet = RegCloseKey(lKey)
End Sub

Private Sub Command1_Click()
   Dim sPath As String, sAppName As String

   sPath$ = App.Path & "\" & "test.exe"
   sAppName = Text1
   If sAppName = "" Then
           MsgBox "Sie muessen einen neuen Namen eigeben!", vbOKOnly + vbCritical, "Fehler...", 0, 0
           Exit Sub
   End If
   Call sSaveApplicationValues(HKEY_LOCAL_MACHINE, "Software\Microsoft\Windows\CurrentVersion\Run", sAppName, sPath)
End Sub

Private Sub Command2_Click()
   Unload Me
End Sub
```

IN MODUL:

```
Declare Function RegCloseKey Lib "advapi32.dll" (ByVal Hkey As Long) As Long

Declare Function RegSetValueEx Lib "advapi32.dll" _
                                Alias "RegSetValueExA" _
                                (ByVal Hkey As Long, _
                                ByVal lpValueName As String, _
                                ByVal Reserved As Long, _
                                ByVal dwType As Long, _
                                lpData As Any, _
                                ByVal cbData As Long) _
                                As Long

Declare Function RegCreateKey Lib "advapi32.dll" _
                                Alias "RegCreateKeyA" _
                                (ByVal Hkey As Long, _
                                ByVal lpSubKey As String, _
                                phkResult As Long) _
```

```
                                As Long

Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const REG_SZ = 1
Public Const REG_DWORD = 4
```

Tipp : 4.

```
'Beispiel : Startparameter in der Registrierdatenbank speichern.
'Standard-Speicherort :
'HKEY_CURRENT_USER\Software\VB and VBA Program Settings\KeyAppName

Private Sub Command2_Click()
   Label1(0) = GetSetting("KeyAppName", "sKeyProgValues", "KeyValState", "0")
   Label1(1) = GetSetting("KeyAppName", "sKeyProgValues", "KeyValHeight", "2730")
   Label1(2) = GetSetting("KeyAppName", "sKeyProgValues", "KeyValLeft", "500")
   Label1(3) = GetSetting("KeyAppName", "sKeyProgValues", "KeyValTop", "500")
   Label1(4) = GetSetting("KeyAppName", "sKeyProgValues", "KeyValWidth", "4770")
End Sub

Private Sub Form_Load()
   Me.WindowState = GetSetting("KeyAppName", "sKeyProgValues", "KeyValState", "0")
   Me.Height = GetSetting("KeyAppName", "sKeyProgValues", "KeyValHeight", "2730")
   Me.Left = GetSetting("KeyAppName", "sKeyProgValues", "KeyValLeft", "500")
   Me.Top = GetSetting("KeyAppName", "sKeyProgValues", "KeyValTop", "500")
   Me.Width = GetSetting("KeyAppName", "sKeyProgValues", "KeyValWidth", "4770")
End Sub

Private Sub Form_Unload(Cancel As Integer)
   SaveSetting "KeyAppName", "sKeyProgValues", "KeyValState", Me.WindowState
   SaveSetting "KeyAppName", "sKeyProgValues", "KeyValHeight", Me.Height
   SaveSetting "KeyAppName", "sKeyProgValues", "KeyValLeft", Me.Left
   SaveSetting "KeyAppName", "sKeyProgValues", "KeyValTop", Me.Top
   SaveSetting "KeyAppName", "sKeyProgValues", "KeyValWidth", Me.Width
End Sub

Private Sub Command1_Click()
   Unload Me
End Sub
```

Tipp : 5.

```
'Beispiel ComboBox fuellen

Dim sarrValue(5) As String

Private Sub Combo1_Click()
        Select Case Combo1.ListIndex
                Case 0, 1
                        Text1.Text = "Testausgabe fuer Index 0 , 1"
                Case 2, 3
                        Text1.Text = "Testausgabe fuer Index 2 , 3"
                Case 4
                        Text1.Text = sarrValue(5) 'Test fuer Index 4
                Case 5
                        Text1.Text = "Testausgabe fuer Index 5"
        End Select
End Sub

Private Sub Command1_Click()
        MsgBox Combo1.Text, vbOKOnly + vbInformation, "Combobox Beispiel"
End Sub

Private Sub Command2_Click()
        Unload Me
End Sub

Private Sub Form_Load()
        Dim I       As Integer

        sarrValue(0) = "Heinz Prelle"
        sarrValue(1) = "Angelika Prelle"
        sarrValue(2) = "Alexandra Prelle"
        sarrValue(3) = "Fritz der Hund"
```

```
        sarrValue(4) = "http://www.vb-5.de"
        sarrValue(5) = "http://www.Visual-Basic5.de"
        For I = 0 To 5
                Combo1.AddItem sarrValue(I)
        Next
        Combo1.Text = sarrValue(0)
End Sub
```

Tipp : 6.

```
'Beispiel : Caption lesen...

Private Declare Function GetWindowTextLength Lib "user32" _
                                Alias "GetWindowTextLengthA" _
                                (ByVal hWnd As Long) _
                                As Long

Private Declare Function GetWindowText Lib "user32" _
                                Alias "GetWindowTextA" _
                                (ByVal hWnd As Long, _
                                ByVal lpString As String, _
                                ByVal cch As Long) _
                                As Long

Function fGetWndTitle(hWnd)
        iRet = GetWindowTextLength(hWnd)
        sBuffer$ = String$(iRet, 0)
        lRet% = GetWindowText(hWnd, sBuffer$, (iRet + 1))
        fGetWndTitle = sBuffer$
End Function

Private Sub Command1_Click()
        Frame1.Caption = fGetWndTitle(Form1.hWnd)
End Sub

Private Sub Command2_Click()
         Unload Me
End Sub
```

Tipp : 7.

```
'Beispiel : Programmeintrag im Kontextmenue und Anzeige von
'gaengigen Grafikformaten auf einen Klick des Kontextmenues.
'Speicherort der Programmregistrierung:
'HKEY_CLASSES_ROOT\*\shell\Prelles-Grafikanzeige
'Vorgehensweise:
'Exe ueber doppelclick starten : das Programm wird registriert
'Im Explorer mit der rechten Mousetaste auf eine sFile klicken
'und dann den Kontextmenueeintrag Prelles-Grafikanzeige waehlen.

Private Declare Function RegCreateKey Lib "advapi32.dll" _
                                Alias "RegCreateKeyA" _
                                (ByVal hKey As Long, _
                                ByVal lpSubKey As String, _
                                phkResult As Long) _
                                As Long

Private Declare Function RegSetValue Lib "advapi32.dll" _
                                Alias "RegSetValueA" _
                                (ByVal hKey As Long, _
                                ByVal lpSubKey As String, _
                                ByVal dwType As Long, _
                                ByVal lpData As String, _
                                ByVal cbData As Long) _
                                As Long

Private Declare Function FindExecutable Lib "shell32.dll" _
                                Alias "FindExecutableA" _
                                (ByVal lpFile As String, _
                                ByVal lpDirectory As String, _
                                ByVal lpResult As String) _
                                As Long

Dim iFileNumber%
```

```
Public Sub sRegisteredApplication()
        Const HKEY_CLASSES_ROOT = &H80000000
        Const REG_SZ = 1

        sMenuString$ = "Prelles-Grafikanzeige"
        sKey$ = "*\shell\" + sMenuString$ + "\command"
        lRet& = RegCreateKey(HKEY_CLASSES_ROOT, sKey$, phkResult&)
        sTmp$ = App.Path + "\" + App.EXEName + ".exe %1"
        lRet& = RegSetValue(HKEY_CLASSES_ROOT, sKey$, REG_SZ, sTmp$, Len(sTmp$))
End Sub

Private Sub Command1_Click()
        Close iFileNumber%
        Unload Me
        End
End Sub

Private Sub Form_Load()
        Me.Caption = "www.Visual-Basic5.de"
         sFileName$ = Command$
         If sFileName$ = "" Then
                sRegisteredApplication
                MsgBox "Es wurde kein sFilename angegeben.", vbOKOnly + _
                vbInformation, "Programm wurde registriert..."
                End
         End If

        On Error Resume Next
        iLen% = Len(Dir$(sFileName$))
        If Err Then
                MsgBox "Kein gültiger sFilename: " + vbCrLf + sFileName$
               End
         End If

        iLen$ = Space$(220)
        lRet& = FindExecutable(sFileName$, lpDirectory$, iLen$)
        iInLen% = InStr(iLen$, Chr$(0))
        If iInLen% Then iLen$ = Left$(iLen$, iInLen% - 1)
        If Len(iLen$) = 0 Then iLen$ = sFileName$
        iFileNumber% = FreeFile
        Open sFileName$ For Input As iFileNumber%
        Image1.Picture = LoadPicture(sFileName$)
End Sub

Private Sub Form_Unload(Cancel As Integer)
        Close iFileNumber%
End Sub
```

Tipp : 8.

```
'Beispiel : Inhalt aus ListBox speichern... / oeffnen...
'Hinweis  :
'Bevor Sie die in diesem Beispiel angegebene sFile oeffnen
'koennen muessen Sie sie speichern.

Function fSaveListContent(sFileName$, obj As ListBox)
   If sFileName$ Like "[{};^&*%]" Then Exit Function
   Open sFileName$ For Output As #1
   For i = 0 To obj.ListCount - 1
   sWriteValue$= obj.List(i)
   Write #1, sWriteValue$
   Next i
   Write #1, " "
   Close #1
End Function

Private Sub Command1_Click()
   Dim sPath$

   sPath$ = App.Path & "\" & "test.txt"
   Call fSaveListContent(sPath$, List1)
   Command3.Enabled = True
End Sub

Private Sub Command2_Click()
```

```
    Unload Me
End Sub

Private Sub Command3_Click()
    Dim sPath$

    sPath$ = App.Path & "\" & "test.txt"
    Call fOpenFile(sPath$, List2)
End Sub

Private Sub Form_Load()
    For i = 0 To 6
            List1.AddItem "Eintrag : " & i
    Next i
End Sub

Function fOpenFile(sFileName$, obj As ListBox)
    List$ = obj.Text
    On Error GoTo 1
    Open sFileName$ For Input As #1
    obj.Clear
    Do
    Input #1, List$
            obj.AddItem List$
            If List$ = " " Then Exit Do
    Loop
1:  Close #1
    Exit Function
    Close #1
End Function
```

Tipp : 9.

```
'Beispiel : System - Werte ausgeben

Dim iWidth As Integer, iHeight As Integer

Const SM_CXSCREEN = 0
Const SM_CYSCREEN = 1

Private Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long

Sub sSysValues(ByRef iRetMaxWidth As Integer, ByRef iRetMaxHeight As Integer)
    iRetMaxWidth= GetSystemMetrics(SM_CXSCREEN)
    iRetMaxHeight= GetSystemMetrics(SM_CYSCREEN)
End Sub

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Command2_Click()
    sSysValues iWidth, iHeight
    Frame1.Caption = "Screen Weite:  " & iWidth
    Frame2.Caption = "Screen Hoehe: " & iHeight
End Sub
```

Tipp : 10.

```
'Beispiel : Text Boxen zur Laufzeit erzeugen und wieder entfernen.

Dim n As Integer

Private Sub Command1_Click()
            Unload Me
End Sub

Private Sub Command2_Click()
            If n < 5 Then
                        n = n + 1
                        Load Text1(n)
                        Text1(n).Top = Text1(n - 1).Top + _
                        Text1(n - 1).Height + 100
                        Text1(n).Text = "TextBox NR.: " & Str(n)
```

```
                Text1(n).Visible = True
                Me.Refresh
        End If
End Sub

Private Sub Command3_Click()
        If n > 0 Then
                Text1(n).Visible = False
                Unload Text1(n)
                n = n - 1
        End If
End Sub
```

Tipp : 11.

```
'Beispiel : Mauszeiger ändern.

Private Sub Form_Load()
   Combo1.AddItem vbArrow: Combo1.AddItem vbCrosshair
   Combo1.AddItem vbIbeam: Combo1.AddItem vbIconPointer
   Combo1.AddItem vbSizePointer: Combo1.AddItem vbSizeNESW
   Combo1.AddItem vbSizeNS: Combo1.AddItem vbSizeNWSE
   Combo1.AddItem vbSizeWE: Combo1.AddItem vbUpArrow
   Combo1.AddItem vbHourglass: Combo1.AddItem vbNoDrop
   Combo1.AddItem vbArrowHourglass: Combo1.AddItem vbArrowQuestion
   Combo1.AddItem vbSizeAll
   Combo1.ListIndex = 0
End Sub

Private Sub Command1_Click()
   Unload Me
   End
End Sub

Sub sChangeMousePointer(mpPointer)
   Dim i As Long
   Me.MousePointer = mpPointer
   For i = 1 To 100000
        DoEvents
   Next
   Me.MousePointer = vbDefault
End Sub

Private Sub Command2_Click()
   sChangeMousePointer (Combo1.Text)
End Sub
```

Tipp : 12.

```
IN MODUL:
String in Systemmenue einguegen.

Declare Function GetSystemMenu Lib "user32" _
                                (ByVal hWnd As Long, _
                                ByVal bRevert As Long) _
                                As Long

Declare Function AppendMenu Lib "user32" _
                                Alias "AppEndMenuA" _
                                (ByVal hMenu As Long, _
                                ByVal wFlags As Long, _
                                ByVal wIDNewItem As Long, _
                                ByVal lpNewItem As Any) _
                                As Long

Declare Function InsertMenu Lib "user32" Alias _
                                "InsertMenuA" _
                                (ByVal hMenu As Long, _
                                ByVal nPosition As Long, _
                                ByVal wFlags As Long, _
                                ByVal wIDNewItem As Long, _
                                ByVal lpNewItem As Any) _
                                As Long
```

```
Const MF_BYPOSITION = &H400&
Const MF_STRING = &H0&
Const WM_SYSCOMMAND = &H112

Declare Function CallWindowProc Lib "user32" Alias _
                                "CallWindowProcA" _
                                (ByVal lpPrevWndFunc As Long, _
                                ByVal hWnd As Long, _
                                ByVal Msg As Long, _
                                ByVal wParam As Long, _
                                ByVal lParam As Long) _
                                As Long

Declare Function SetWindowLong Lib "user32" Alias _
                                "SetWindowLongA" _
                                (ByVal hWnd As Long, _
                                ByVal nIndex As Long, _
                                ByVal dwNewLong As Long) _
                                As Long

Const GWL_WNDPROC = -4
Dim lWndProcess   As Long
Dim frm                         As Form

Public Sub sHookForm(frm As Form)
   If lWndProcess Then
            sUnHookForm
   End If
   Set frm = frm
   lWndProcess = SetWindowLong(frm.hWnd, _
   GWL_WNDPROC, _
   AddressOf NewWndProcess)
End Sub

Public Sub sUnHookForm()
   If lWndProcess = 0 Then Exit Sub
   Dim lRet            As Long

   lRet = SetWindowLong(frm.hWnd, _
   GWL_WNDPROC, lWndProcess)
   lWndProcess = 0
   Set frm = Nothing
End Sub

Function NewWndProcess(ByVal hWnd As Long, ByVal lMessage As Long, ByVal wParam As Long, _
                                                                        ByVal lParam As Long)
                                                                        As Long
   If lMessage = WM_SYSCOMMAND Then
            If (wParam And &HF000) = 0 Then
                     frm.sSysMenu wParam
            End If
   End If
   NewWndProcess = CallWindowProc(lWndProcess, _
   hWnd, lMessage, wParam, lParam)
End Function

Sub sAddStringToSysMenue(sString$, lNumber&&, Optional vPos)
lRet&= GetSystemMenu(frm.hWnd, False)
   If IsMissing(vPos) Then
            lRet& = AppEndMenu(lngMenue&, MF_STRING, lNumber&, sString$)
   Else
            lRet& = InsertMenu(lngMenue&, vPos, MF_STRING Or MF_BYPOSITION, _
            lNumber&, sString$)
   End If
End Sub

IN FORM:

Const SYSMENULINE = 1100

Private Sub Command1_Click()
   sUnHookForm
   Unload Me
End Sub

Private Sub Form_Load()
   sHookForm Me
```

```
    sAddStringToSysMenue "&Über www.Visual-Basic5.de", SYSMENULINE
End Sub

Private Sub Form_Unload(Cancel As Integer)
    sUnHookForm
End Sub

Sub sSysMenu(lNumber&)
    Select Case lNumber&
    Case SYSMENULINE
            MsgBox "Free Download bei www.Visual-Basic5.de", vbOKOnly + vbInformation, "Meldung...", 0, 0
    End Select
End Sub
```

Tipp : 13.

```
'Beispiel : Sortierte Root Ausgabe:

Dim sFileName As String, sFolderName As String
Dim sarrFolder() As String, sarrFile() As String
Dim iAllFolder As Integer, iAllFiles As Integer
Dim n As Integer

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Command2_Click()
    List1.Clear
    List2.Clear
    sFolderName = Text1
    sFileName = Dir(sFolderName, vbNormal + vbDirectory)
    Do While sFileName <> ""
    If sFileName <> "." And sFileName <> ".." Then
            If (GetAttr(sFolderName & sFileName) And vbDirectory) = vbDirectory Then
                    iAllFolder = iAllFolder + 1
                    ReDim Preserve sarrFolder(iAllFolder)
                    sarrFolder(iAllFolder) = sFileName
             Else
                    iAllFiles = iAllFiles + 1
                    ReDim Preserve sarrFile(iAllFiles)
                    sarrFile(iAllFiles) = sFileName
                    End If
            End If

    sFileName = Dir
    Loop
            sQuickSort sarrFolder, 0, UBound(sarrFolder)
            sQuickSort sarrFile, 0, UBound(sarrFile)
            For n = 0 To UBound(sarrFolder)
                    List1.AddItem sarrFolder(n)
            Next
            For n = 0 To UBound(sarrFile)
                    List2.AddItem sarrFile(n)
            Next
End Sub

Public Sub sQuickSort(varrSorted As VarilRet, ByVal iFirst As Integer, ByVal iLast As Integer)

    Dim iDown As Integer, iHeight As Integer
    Dim vTmp As VarilRet, vCut As VarilRet
    iDown = iFirst
    iHeight = iLast
    vCut = UCase(varrSorted((iFirst + iLast) / 2))
    Do
            Do While (UCase(varrSorted(iDown)) < vCut)
                     iDown = iDown + 1
             Loop
             Do While (UCase(varrSorted(iHeight)) > vCut)
                    iHeight = iHeight - 1
             Loop
    If (iDown <= iHeight) Then
            vTmp = varrSorted(iDown)
            varrSorted(iDown) = varrSorted(iHeight)
            varrSorted(iHeight) = vTmp
            iDown = iDown + 1
```

```
            iHeight = iHeight - 1
    End If

    Loop While (iDown <= iHeight)

    If (iFirst < iHeight) Then sQuickSort varrSorted, iFirst, iHeight
    If (iDown < iLast) Then sQuickSort varrSorted, iDown, iLast
End Sub
```

Tipp : 14.

```
'Beispiel : Rekursive Sortierung.
'Eingabebeispiel : ABCD oder 1234

Private sPermutation            As String
Private sChar                   As String
Private iPos()                  As Integer
Private iPosPtr                 As Integer

Private Sub Command2_Click()
    Unload Me
End Sub

Private Sub Command1_Click()
    Call sPermutationRecursive(Text1.Text)
    Command1.Enabled = False
    Command3.Enabled = True
    sCountSortValues
End Sub

Public Sub sPermutationRecursive(sValue As String)
    sChar = sValue
    iPosPtr = -1
    ReDim iPos(Len(sChar) - 1)
    Call sSort(0)
End Sub

Private Sub sSort(iPointerPosition As Integer)
            Dim i       As Integer

            iPosPtr = iPosPtr + 1
            iPos(iPointerPosition) = iPosPtr

            If iPosPtr = Len(sChar) Then
                        sPermutation = ""
                        For i = 0 To UBound(iPos)
                                    sPermutation = sPermutation & Mid$(sChar, iPos(i), 1)
                        Next i
                                    List1.AddItem sPermutation
            Else
                        For i = 0 To Len(sChar) - 1
                        If iPos(i) = 0 Then Call sSort(i)
            Next i
            End If
            iPosPtr = iPosPtr - 1
            iPos(iPointerPosition) = 0
End Sub

Sub sCountSortValues()
            sRes  = List1.ListCount
            Frame2.Caption = sRes & " Sortiermoeglichkeiten"
End Sub

Private Sub Command3_Click()
            List1.Clear
            Command1.Enabled = True
            Command3.Enabled = False
End Sub

Private Sub Text1_Change()
            If Text1 = vbNullString Then Text1 = "A"
End Sub
```

Tipp : 15.

```
'Beispiel : Cursor Position an bestimmten Punkt setzen.
'Hinweis  : ScaleMode der Form ist vbPixels [3]

Private Type POINTAPI
        x      As Long
        y      As Long
End Type

Private Declare Function ClientToScreen Lib "user32" _
                                        (ByVal hwnd As Long, _
                                        lpPoint As POINTAPI) _
                                        As Long

Private Declare Function SetCursorPos Lib "user32" _
                                        (ByVal x As Long, _
                                        ByVal y As Long) _
                                        As Long

Public Function fSetCursorPos(obj As Object, lLeft As Long, lTop As Long) As Boolean
    On Error GoTo Fehler

    Dim x As Long, y As Long
    Dim lRet As Long, lHandle As Long
    Dim pa As POINTAPI

    lHandle = obj.hwnd

    With Screen
            x = CLng(lLeft / .TwipsPerPixelX)
            y = CLng(lTop / .TwipsPerPixelY)
    End With

    pa.x = x
    pa.y = y
    lRet = ClientToScreen(lHandle, pa)
    lRet = SetCursorPos(pa.x, pa.y)
    fSetCursorPos = (lRet <> 0)
    Exit Function
ErrHandle:
    fSetCursorPos = False
    Exit Function
End Function

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Command2_Click()
    'fSetCursorPos Me, Me.Width / 2, Me.Height / 2
    fSetCursorPos Me, 200, 200
End Sub
```

Tipp : 16.

```
'Beispiel : "ToolTipp - Text" in einer Listbox anzeigen.

Private Declare Function SendMessage Lib "user32" _
                                        Alias "SendMessageA" _
                                        (ByVal hwnd As Long, _
                                        ByVal wMsg As Long, _
                                        ByVal wParam As Long, _
                                        lParam As Any) _
                                        As Long

Private Const LB_ITEMFROMPOINT = &H1A9

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    For i = 1 To 10
            List1.AddItem "www.Visual-Basic5.de " & i
    Next i
```

```
End Sub

Private Sub List1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim lLeft         As Long
   Dim lTop          As Long
   Dim lIndex        As Long

   If Button = 0 Then
            lLeft = CLng(X / Screen.TwipsPerPixelX)
            lTop = CLng(Y / Screen.TwipsPerPixelY)
             With List1
                        lIndex = SEndMessage(.hwnd, LB_ITEMFROMPOINT, 0, ByVal ((lTop * 65536) + lLeft))
                        If (lIndex >= 0) And (lIndex <= .ListCount) Then
                                    .ToolTipText = .List(lIndex)
                        Else
                                    .ToolTipText = ""
                        End If
                        End With
            End If
End Sub
```

Tipp : 17.

```
'Beispiel : sGradientForm Horizontal ROT > SCHWARZ

Sub sGradientForm(obj As Form)
   Dim n As Integer, iRed As Integer, iGreen As Integer, iBlue As Integer

   obj.DrawStyle = vbInsideSolid
   obj.DrawMode = vbCopyPen
   obj.ScaleMode = vbPixels
   obj.DrawWidth = 2
   MousePointer = 11

   iRed = 255
   iGreen = 0
   iBlue = 0

   DoEvents

   obj.ScaleHeight = 256
   For n = 1 To 255
            iRed = 255 - n
            obj.Line (0, n)-(Screen.Height, n - 1), _
            RGB(iRed, iGreen, iBlue), B
   Next n

   MousePointer = 0
End Sub

Private Sub Command1_Click()
   End
End Sub

Private Sub Form_Activate()
      sGradientForm Me
End Sub
```

Tipp : 18.

```
'Beispiel : Auswahl-Box zeichnen...

Dim ix1 As Integer, ix2 As Integer
Dim iy1 As Integer, iy2 As Integer

Private Sub Command1_Click()
   Unload Me
End Sub

Private Sub Picture1_MouseDown(Button As Integer, ShIft As Integer, X As Single, Y As Single)
   Picture1.DrawMode = 6
   Picture1.DrawStyle = 2
   If ix2 > 0 Then Picture1.Line (ix1, iy1)-(ix2, iy2), , B
   ix1 = X
```

```
    iy1 = Y
    ix2 = X
    iy2 = Y
End Sub

Private Sub Picture1_MouseMove(Button As Integer, ShIft As Integer, X As Single, Y As Single)
    If Button = 1 Then
            Picture1.Line (ix1, iy1)-(ix2, iy2), , B
            ix2 = X
            iy2 = Y
            Picture1.Line (ix1, iy1)-(X, Y), , B
    End If
End Sub
```

Tipp : 19.

IN FORM:

```
'Beispiel : Explorer W95 etc. aendern 1 i-net - Version...
'Diese Version funktioniert nur wenn der Explorer vor dem
'Beispiel geladen wurde...

Private Sub Command1_Click()
            Unload Me
End Sub

Private Sub Form_Load()
  Move (Screen.Width - Width) * 0.5, (Screen.Height - Height) * 0.5
  Call sShellView(GetDesktopWindow)
  If List1.ListCount Then List1.ListIndex = 0
End Sub

Private Sub sShellView(lhwnd As Long)
    Dim lChildHwnd  As Long
    Dim lShellHwnd  As Long

    lChildHwnd = GetWindow(lhwnd, clbHwnd)

    Do While lChildHwnd
            lShellHwnd = FindWindowEx(lChildHwnd, 0, WC_SHVIEW, vbNullString)
            If lShellHwnd Then
                    List1.AddItem "0x" & Hex(lShellHwnd) & vbTab & fStingValue(fTopLevelPointer(lShellHwnd))
                    List1.ItemData(List1.NewIndex) = lShellHwnd
            Else
                    Call sShellView(lChildHwnd)
            End If
    lChildHwnd = GetWindow(lChildHwnd, clbZero)
    Loop
End Sub

Private Sub cmdView_Click()
            Dim lShellHwnd      As Long
            Dim lId             As Long

             lShellHwnd = List1.ItemData(List1.ListIndex)
            If IsWindow(lShellHwnd) Then

            Select Case True
                    Case Option1(0): lId = IDM_SHVIEW_LARGEICON
                    Case Option1(1): lId = IDM_SHVIEW_SMALLICON
                    Case Option1(2): lId = IDM_SHVIEW_LIST
                    Case Option1(3): lId = IDM_SHVIEW_REPORT
            End Select
            Call SendMessage(lShellHwnd, WM_COMMAND, lId, 0)
            Else
                    MsgBox "Falsches ID [Fehler...]", vbInformation
                    List1.Clear
                    Call sShellView(GetDesktopWindow)
                    If List1.ListCount Then List1.ListIndex = 0
            End If
End Sub
```

IN MODUL:

```
Option Explicit
Declare Function SendMessage Lib "user32" _
```

```
                                Alias "SendMessageA" _
                                (ByVal hWnd As Long, _
                                ByVal wMsg As Long, _
                                ByVal wParam As Long, _
                                lParam As Any) _
                                As Long

Public Const WM_COMMAND = &H111

Public Const IDM_SHVIEW_LARGEICON = &H7029
Public Const IDM_SHVIEW_SMALLICON = &H702A
Public Const IDM_SHVIEW_LIST = &H702B
Public Const IDM_SHVIEW_REPORT = &H702C

Public Const IDM_SHVIEW_TOGGLECOL0 = &H7230
Public Const IDM_SHVIEW_TOGGLECOL1 = &H7231
Public Const IDM_SHVIEW_TOGGLECOL2 = &H7232
Public Const IDM_SHVIEW_TOGGLECOL3 = &H7233

Public Const WC_SHVIEW = "SHELLDLL_DefView"
Public Const WC_LISTVIEW = "SysListView32"

Public Enum IconView
  LVS_ICON = &H0
  LVS_REPORT = &H1
  LVS_SMALLICON = &H2
  LVS_LIST = &H3
  LVS_TYPEMASK = &H3
End Enum

Declare Function GetParent Lib "user32" (ByVal hWnd As Long) As Long

Declare Function FindWindowEx Lib "user32" _
                                Alias "FindWindowExA" _
                                (ByVal lhwnd As Long, _
                                ByVal hwndChildAfter As Long, _
                                ByVal lpszClass As String, _
                                ByVal lpszWindow As String) _
                                As Long

Declare Function GetWindowText Lib "user32" Alias _
                                "GetWindowTextA" (ByVal hWnd As Long, _
                                ByVal lpString As String, _
                                ByVal cch As Long) _
                                As Long

Declare Function IsWindow Lib "user32" (ByVal hWnd As Long) _
                                As Long

Declare Function GetDesktopWindow Lib "user32" () As Long

Declare Function GetWindow Lib "user32" (ByVal hWnd As Long, _
                                ByVal wCmd As ListCommand) _
                                As Long

Public Enum ListCommand
  lcWnd0Entry = 0
  lcWnd1Entry = 1
  clbZero = 2
  lcWndPrev = 3
  lcOwner = 4
  clbHwnd = 5
  lcMax = 5
End Enum

Public Declare Function GetWindowLong Lib "user32" Alias _
                                "GetWindowLongA" _
                                (ByVal hWnd As Long, _
                                ByVal nIndex As Long) _
                                As Long

Public Const GWL_STYLE = (-16)

Public Function fTopLevelPointer(hWnd As Long) As Long
    Dim lhwnd       As Long
    Dim lTmp        As Long
    lhwnd = hWnd
```

```
    Do
        lTmp = GetParent(lhwnd)
        If lTmp Then lhwnd = lTmp
    Loop While lTmp
    fTopLevelPointer = lhwnd
End Function

Public Function fStingValue(hWnd As Long) As String
    Dim sBuffer As String * 260

    If GetWindowText(hWnd, sBuffer, 260) Then
        fStingValue = Left$(sBuffer, InStr(sBuffer, vbNullChar) - 1)
    End If
End Function
```

Tipp : 20.

```
'Beispiel : Grundfunktion sFileen anzeigen...Standard

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Command2_Click()
    List1.Clear
    Dim sFileName   As String

    sFileName = Dir(Text1, vbNormal + vbDirectory)
    Do While sFileName <> ""
        List1.AddItem sFileName
        sFileName = Dir
    Loop
End Sub
```

Tipp : 21.

```
'Beispiel : Kompletten sPath in seine Einzelkomponenten
'zerlegen...

Private Sub sCutPath(ByVal sPath As String, ByRef sDrive As String, ByRef sSubFolder As String, _
                                                    ByRef sFileName As String, ByRef sExtension As String)
    Dim lLastPoint    As Long
    Const cBackSlash = "\"

    sDrive = Left(sPath, 3)
    lLastPoint = fGetCharVal(sPath, cBackSlash)
    sSubFolder = Mid(sPath, 4, lLastPoint - 4)
    sFileName = Mid(sPath, lLastPoint, Len(sPath) - lLastPoint - 3)
    sExtension = Right(sPath, 4)
End Sub

Function fGetCharVal(ByVal sSearchString As String, ByVal sSearchWath As String) As Long
    Dim iCurrPosition As Integer, iNextPosition As Integer

    iNextPosition = 0
    Do
        iCurrPosition = iNextPosition
        iNextPosition = InStr(iCurrPosition + 1, sSearchString, sSearchWath)
    Loop Until iNextPosition = 0
    fGetCharVal = iCurrPosition + 1
End Function

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Command2_Click()
        Dim sPathName As String
        Dim sFolder             As String
        Dim sSubFolder          As String
        Dim sFile               As String
        Dim sSufix              As String

        sPathName = App.Path & "\" & "TestsFile.txt"
```

```
        sCutPath sPathName, strVerzeichnis, sSubFolder, sFile, sSufix
        Frame1(0).Caption = sPathName
        Frame1(1).Caption = strVerzeichnis
        Frame1(2).Caption = sSubFolder
        Frame1(3).Caption = sFile
        Frame1(4).Caption = sSufix
End Sub

Private Sub Form_Load()
        Frame1(0).Caption = "Kompletter Pfad : "
        Frame1(1).Caption = "LW : "
        Frame1(2).Caption = "sPath ohne LW : "
        Frame1(3).Caption = "sFilename ohne Endung : "
        Frame1(4).Caption = "Endung : "
End Sub
```

Tipp : 22.

```
'Beispiel : Freien Festplattenplatz ermitteln...
'Hinweis  : Auf eine Fehlerbehandlung der TB wurde verzichtet.

Private Declare Function GetDiskFreeSpace Lib "kernel32" Alias _
                                "GetDiskFreeSpaceA" _
                                (ByVal lpRootPathName As String, _
                                lpSectorsPerCluster As Long, _
                                lpBytesPerSector As Long, _
                                lpNumberOfFreeClusters As Long, _
                                lpTotalNumberOfClusters As Long) _
                                As Long

Dim sCurrDrive As String

Function fGetFreeDiskSpace(ByVal sDrv As String) As Long
  Dim lFreeSpace            As Long
  Dim lCluster              As Long
  Dim lBytesPerSector       As Long
  Dim lFreeCluster          As Long
  Dim lTotalCluster         As Long

   If Err <> 0 Then
        lFreeSpace = -1
   Else
        lFreeSpace = GetDiskFreeSpace(sDrv, lCluster, _
        lBytesPerSector, lFreeCluster, lTotalCluster)
   If lFreeSpace <> 1 Then
        lFreeSpace = -1
   Else
        lFreeSpace = lCluster * lBytesPerSector * lFreeCluster
   End If
   End If
   If lFreeSpace = -1 Then
        MsgBox "Fehler..."
   End If
   fGetFreeDiskSpace = lFreeSpace
   Frame2.Caption = "Rest auf LW : " & sCurrDrive & " " & Format(fGetFreeDiskSpace, "##,##")
   Err = 0
End Function

Private Sub Command1_Click()
   Unload Me
End Sub

Private Sub Command2_Click()
   sCurrDrive = Text1
   fGetFreeDiskSpace (sCurrDrive)
End Sub
```

Tipp : 23.

In Form:

```
'Beispiel : Host / IP Ermittlung...
```

```
Option Explicit

Private Sub Command3_Click()
   Text1 = fGetHostName()
   Text2 = fGetIpAdress()
End Sub

Private Sub Command1_Click()
   Unload Me
End Sub

Public Function fSaveFile(sPath As String, sValue As String)
   Dim iFileNumber As Integer

   iFileNumber = FreeFile
         Open sPath For Append As #iFileNumber
         Print #iFileNumber, sValue
   Close #iFileNumber
End Function

Private Sub Command2_Click()
   Dim sPath        As String

   sPath = App.Path & "\" & "ip.log"
   fSaveFile sPath, Text2
End Sub

Private Sub Text2_Change()
   Command2.Enabled = True
End Sub
```

IN MODUL:

```
Option Explicit

Public Const IPDESCRIBE = 256
Public Const IPSYSSTATE = 128
Public Const IPERROR              As Long = 0
Public Const IPVERREQUEST         As Long = &H101
Public Const IPMAJORVERSION       As Long = IPVERREQUEST \ &H100 And &HFF&
Public Const IPMINORVERSION       As Long = IPVERREQUEST And &HFF&
Public Const IPSOCKETREQUEST      As Long = 1
Public Const IPSOCKETERR          As Long = -1

Public Type HostValue
   hvName         As Long
   hvAlias        As Long
   hvAdressType   As Integer
   hvLen          As Integer
   hvList         As Long
End Type

Public Type DataValues
   dvVersion                                  As Integer
   dvHeightVersion                            As Integer
   dvDescription(0 To IPDESCRIBE)             As Byte
   dvSysState(0 To IPSYSSTATE)                As Byte
   dvMaxState                                 As Integer
   dvMaxUpdate                                As Integer
   dvVendorInf                                As Long
End Type

Public Declare Function WSAGetLastError Lib _
                        "WSOCK32.DLL" () _
                        As Long

Public Declare Function WSAStartup Lib "WSOCK32.DLL" _
                        (ByVal wVersionRequired As Long, _
                        lpWSADATA As DataValues) _
                        As Long

Public Declare Function WSACleanup Lib _
                        "WSOCK32.DLL" () As Long

Public Declare Function gethostname Lib "WSOCK32.DLL" _
                        (ByVal szHost As String, _
                        ByVal dwHostLen As Long) _
```

```
                                As Long

Public Declare Function gethostbyname Lib "WSOCK32.DLL" _
                                (ByVal szHost As String) _
                                As Long

Public Declare Sub CopyMemory Lib "kernel32" _
                                Alias "RtlMoveMemory" (hpvDest As Any, _
                                ByVal hpvSource As Long, _
                                ByVal cbCopy As Long)

Public Function fGetIpAdress() As String
   Dim sHstName             As String * 256
   Dim lHst                 As Long
   Dim hvHost               As HostValue
   Dim lIpAdress            As Long
   Dim barrIp()             As Byte
   Dim n                    As Integer
   Dim sRes                 As String

   If Not fInitSocket() Then
            fGetIpAdress = ""
            Exit Function
   End If

   If gethostname(sHstName, 256) = IPSOCKETERR Then
            fGetIpAdress = ""
            MsgBox "Window Socket Fehler " & Str$(WSAGetLastError()) & " entdeckt. Host Name ist nicht zu ermitteln."
            sErr
            Exit Function
   End If

   sHstName = Trim$(sHstName)
   lHst = gethostbyname(sHstName)

   If lHst = 0 Then
            fGetIpAdress = ""
            MsgBox "Window Socket nicht ermittelt. " & "Host Name ist nicht zu ermitteln."
            sErr
            Exit Function
   End If

   CopyMemory hvHost, lHst, Len(hvHost)
   CopyMemory lIpAdress, hvHost.hvList, 4
   ReDim barrIp(1 To hvHost.hvLen)
   CopyMemory barrIp(1), lIpAdress, hvHost.hvLen

   For n = 1 To hvHost.hvLen
            sRes = sRes & barrIp(n) & "."
   Next
            fGetIpAdress = Mid$(sRes, 1, Len(sRes) - 1)
            sErr
End Function

Public Function fGetHostName() As String
   Dim sHstName    As String * 256

   If Not fInitSocket() Then
            fGetHostName = ""
            Exit Function
   End If

   If gethostname(sHstName, 256) = IPSOCKETERR Then
            fGetHostName = ""
            MsgBox "Window Socket Fehler " & Str$(WSAGetLastError()) & " entdeckt.  Host Name ist nicht zu ermitteln."
            sErr
            Exit Function
   End If
            fGetHostName = Left$(sHstName, InStr(sHstName, Chr(0)) - 1)
            sErr
End Function

Public Function fObByte(ByVal iParam As Integer) As Byte
            fObByte = (iParam And &HFF00&) \ (&H100)
End Function

Public Function fInitSocket() As Boolean
```

```
    Dim dv              As DataValues
    Dim suByte          As String
    Dim soByte          As String

    If WSAStartup(IPVERREQUEST, dv) <> IPERROR Then
            MsgBox "Die 32-bit Win Socket ist nicht zu finden."
            fInitSocket = False
            Exit Function
    End If

    If dv.dvMaxState < IPSOCKETREQUEST Then
            MsgBox "Diese Anwendung benoetigt ein Minimum " & CStr(IPSOCKETREQUEST) & " Supported Sockets."
            fInitSocket = False
            Exit Function
    End If

    If fUbByte(dv.dvVersion) < IPMAJORVERSION Or  (fUbByte(dv.dvVersion) = IPMAJORVERSION And _
                                                    fObByte(dv.dvVersion) < IPMINORVERSION) Then
            soByte = CStr(fObByte(dv.dvVersion))
            suByte = CStr(fUbByte(dv.dvVersion))
            MsgBox "Socket Version " & suByte & "." & soByte & " ist nicht Supported bei 32-bit Win Socket."
            fInitSocket = False
            Exit Function
    End If

    fInitSocket = True
End Function

Public Function fUbByte(ByVal iParam As Integer) As Byte
            fUbByte = iParam And &HFF&
End Function

Public Sub sErr()
    If WSACleanup() <> IPERROR Then
            MsgBox "Socket Fehler."
    End If
End Sub
```

Tipp : 24.

```
'Beispiel : Test auf Alphanummerische Uebereinstimmung...
'False = Die Zeichen sind Alphabetisch
'True =  Die Zeichen sind Gemischt also Alphanummerisch..

Function fIsAlphaNumeric(ByVal sString As String) As Boolean
    Dim i                           As Integer
    Dim sCurrChar    As String

    For i = 1 To Len(sString)
            sCurrChar = UCase(Mid(sString, i, 1))
            If (Asc(sCurrChar) < 65 Or Asc(sCurrChar) > 90) And Not IsNumeric(sCurrChar) Then
                    fIsAlphaNumeric = False
                    Exit Function
            End If
    Next

    fIsAlphaNumeric = True
End Function

Private Sub Command1_Click()
            Unload Me
End Sub

Private Sub Command2_Click()
    Dim sString         As String

    sString = Text1
    Frame3.Caption = "Die Zeichenkette aus Text1 " & Text1 & " ist nicht Alphanummerisch : " & _
                     fIsAlphaNumeric(sString)
    sString = Text2
    Frame4.Caption = "Die Zeichenkette aus Text2 " & Text2 & " ist Alphanummerisch : " & _
                     fIsAlphaNumeric(sString)
End Sub
```

Tipp : 25.

```
'Beispiel : Konsolenaufruf

Private Declare Function SetConsoleMode Lib "kernel32" (ByVal hConsoleOutput As Long, _
                                    dwMode As Long) _
                                    As Long

Private Declare Function FreeConsole Lib "kernel32" () As Long

Private Declare Function GetStdHandle Lib "kernel32" (ByVal nStdHandle As Long) As Long

Private Declare Function WriteConsole Lib "kernel32" _
                                    Alias "WriteConsoleA" _
                                    (ByVal hConsoleOutput As Long, _
                                    ByVal lpBuffer As Any, _
                                    ByVal nNumberOfCharsToWrite As Long, _
                                    lpNumberOfCharsWritten As Long, _
                                    lpReserved As Any) _
                                    As Long

Private Declare Function AllocConsole Lib "kernel32" () As Long

Private Declare Function ReadConsole Lib "kernel32" Alias _
                                    "ReadConsoleA" _
                                    (ByVal hConsoleInput As Long, _
                                    ByVal lpBuffer As String, _
                                    ByVal nNumberOfCharsToRead As Long, _
                                    lpNumberOfCharsRead As Long, _
                                    lpReserved As Any) _
                                    As Long

Private Const STD_INPUT_HANDLE = -10&
Private Const STD_OUTPUT_HANDLE = -11&
Private Const STD_ERROR_HANDLE = -12&
Private Const ENABLE_LINE_INPUT = &H2
Private Const ENABLE_ECHO_INPUT = &H4
Private Const ENABLE_MOUSE_INPUT = &H10
Private Const ENABLE_PROCESSED_INPUT = &H1
Private Const ENABLE_WINDOW_INPUT = &H8
Private Const ENABLE_PROCESSED_OUTPUT = &H1
Private Const ENABLE_WRAP_AT_EOL_OUTPUT = &H2

Private Sub Command1_Click()
   Unload Me
End Sub

Private Sub Command2_Click()
   Dim sConsoleInput                    As String
   Dim sInput As Long, sOutput As Long, lErr As Long
   Dim sBuffer                          As String * 255

   AllocConsole

   sInput = GetStdHandle(STD_INPUT_HANDLE)
   sOutput = GetStdHandle(STD_OUTPUT_HANDLE)
   lErr = GetStdHandle(STD_ERROR_HANDLE)
   SetConsoleMode sInput, ENABLE_ECHO_INPUT
   sConsoleInput = "Geben Sie eine Zahl zwischen 1 und 10 ein." & vbCrLf
   WriteConsole sOutput, sConsoleInput, Len(sConsoleInput), vbNull, vbNull
   ReadConsole sInput, sBuffer, Len(sBuffer), vbNull, vbNull
   sBuffer = fTrim(sBuffer)
   If IsNumeric(sBuffer) Then
            If CLng(sBuffer) < 1 Or CLng(sBuffer) > 10 Then
            sConsoleInput = "Sie muessen den Anweisungen folgen..."
   Else
            sConsoleInput = "Ihre Eingabe war : " & sBuffer
   End If
   Else
            sConsoleInput = "Sie muessen den Anweisungen folgen..."
   End If
   sConsoleInput = sConsoleInput & vbCrLf
   WriteConsole sOutput, sConsoleInput, Len(sConsoleInput), vbNull, vbNull
   sConsoleInput = "Druecken Sie ENTER um zu Beenden." & vbCrLf
   WriteConsole sOutput, sConsoleInput, Len(sConsoleInput), vbNull, vbNull
   ReadConsole sInput, sBuffer, Len(sBuffer), vbNull, vbNull
   sBuffer = fTrim(sBuffer)
```

```
    FreeConsole
End Sub

Public Function fTrim(ByVal sInputString As String) As String
    Dim sRes        As String
    Dim sChar       As String
    Dim lLen        As Long
    Dim lTmp        As Long

    sRes = sInputString
    lLen = Len(sInputString)
    If lLen > 0 Then
            For lTmp = 1 To lLen
            sChar = Mid(sRes, lTmp, 1)
                    If Asc(sChar) > 32 Then Exit For
            Next

    sRes = Mid(sRes, lTmp)
    lLen = Len(sRes)
    If lLen > 0 Then
            For lTmp = lLen To 1 Step -1
            sChar = Mid(sRes, lTmp, 1)
                    If Asc(sChar) > 32 Then Exit For
            Next
    End If
    sRes = Left$(sRes, lTmp)
    End If
    fTrim = sRes
End Function
```

Tipp : 26.

```
'Beispiel : Startmenue oeffnen

Private Declare Sub keybd_event Lib "user32" _
                                    (ByVal bVk As Byte, _
                                    ByVal bScan As Byte, _
                                    ByVal dwFlags As Long, _
                                    ByVal dwExtraInfo As Long)

Const VK_CONTROL = &H11
Const KEYEVENTF_KEYUP = &H2
Const VK_ESCAPE = &H1B

Private Sub cmdStart_Click()
    Call keybd_event(VK_CONTROL, 0, 0, 0)
    Call keybd_event(VK_ESCAPE, 0, 0, 0)
    Call keybd_event(VK_ESCAPE, 0, KEYEVENTF_KEYUP, 0)
    Call keybd_event(VK_CONTROL, 0, KEYEVENTF_KEYUP, 0)
End Sub

Private Sub Command1_Click()
    End
End Sub
```

Tipp : 27.

```
'Beispiel : Windows Hilfe aufrufen...

Private Declare Sub keybd_event Lib "user32" _
                                    (ByVal bVk As Byte, _
                                    ByVal bScan As Byte, _
                                    ByVal dwFlags As Long, _
                                    ByVal dwExtraInfo As Long)

Const VK_CONTROL = &H11
Const KEYEVENTF_KEYUP = &H2
Const VK_ESCAPE = &H1B

Private Sub Command1_Click()
    End
End Sub

Private Sub Command2_Click()
```

```
    Call keybd_event(VK_CONTROL, 0, 0, 0)
    Call keybd_event(VK_ESCAPE, 0, 0, 0)
    Call keybd_event(VK_ESCAPE, 0, KEYEVENTF_KEYUP, 0)
    Call keybd_event(VK_CONTROL, 0, KEYEVENTF_KEYUP, 0)
    SendKeys "{H}"
End Sub
```

Tipp : 28.

IN MODUL:
Keyboard und Mouseereignisse.

```
Option Explicit

Public Enum MouseKeyFlags
  mkfMouseDown = 1
  mkfMouseUp = 2
  mkfMouseMove = 4
  mkfKeyDown = 8
  mkfKeyUp = 16
End Enum

Private Declare Function SetWindowsHookEx Lib _
                         "user32" Alias "SetWindowsHookExA" _
                         (ByVal idHook As Long, _
                         ByVal lpfn As Long, _
                         ByVal hmod As Long, _
                         ByVal dwThreadId As Long) _
                         As Long

Private Declare Function CallNextHookEx Lib "user32" _
                         (ByVal lHook As Long, _
                         ByVal lCode As Long, _
                         ByVal wParam As Long, _
                         ByVal lParam As Long) _
                         As Long

Private Declare Function UnhookWindowsHookEx Lib "user32" _
                         (ByVal lHook As Long) _
                         As Long

Private Declare Sub CopyMemory Lib "kernel32" Alias _
                         "RtlMoveMemory" _
                         (lpDest As Any, _
                         lpSource As Any, _
                         ByVal cBytes As Long)

Private Declare Function GetAsyncKeyState% Lib "user32" _
                         (ByVal vKey As Long)

Private Declare Function SetWindowPos Lib "user32" _
                         (ByVal hwnd As Long, _
                         ByVal hWndInsertAfter As Long, _
                         ByVal x As Long, _
                         ByVal y As Long, _
                         ByVal cx As Long, _
                         ByVal cy As Long, _
                         ByVal wFlags As Long) _
                         As Long

Private Const SWP_NOSIZE = &H1
Private Const SWP_NOMOVE = &H2
Private Const SWP_NOREDRAW = &H8

Private Const WM_KEYDOWN = &H100
Private Const WM_KEYUP = &H101
Private Const WM_MOUSEMOVE = &H200
Private Const WM_LBUTTONDOWN = &H201
Private Const WM_LBUTTONUP = &H202
Private Const WM_LBUTTONDBLCLK = &H203
Private Const WM_RBUTTONDOWN = &H204
Private Const WM_RBUTTONUP = &H205
Private Const WM_RBUTTONDBLCLK = &H206
Private Const WM_MBUTTONDOWN = &H207
Private Const WM_MBUTTONUP = &H208
Private Const WM_MBUTTONDBLCLK = &H209
Private Const WM_MOUSEWHEEL = &H20A
```

```
Private Const WH_JOURNALRECORD = 0

Type EventMessage
    wMsg            As Long
    lParamLow       As Long
    lParamHigh      As Long
End Type

Dim em    As EventMessage

Dim lHook As Long, obj As Form, lFlags As Long

Public Function fHookApp(ByVal lCode As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
   If lCode < 0 Then
          fHookApp = CallNextHookEx(lHook, lCode, wParam, lParam)
          Exit Function
   End If

   Dim i%, j%, k%

   CopyMemory em, ByVal lParam, Len(em)
   Select Case em.wMsg
          Case WM_KEYDOWN
                If (lFlags And mkfKeyDown) = mkfKeyDown Then
                      If GetAsyncKeyState(vbKeyShIft) Then j = 1
                      If GetAsyncKeyState(vbKeyControl) Then j = 2
                      If GetAsyncKeyState(vbKeyMenu) Then j = 4
                      k = (em.lParamLow And &HFF)
                      obj.System_KeyDown k, j
                End If
   Case WM_KEYUP
          If (lFlags And mkfKeyUp) = mkfKeyUp Then
                If GetAsyncKeyState(vbKeyShIft) Then j = 1
                If GetAsyncKeyState(vbKeyControl) Then j = 2
                If GetAsyncKeyState(vbKeyMenu) Then j = 4
                k = (em.lParamLow And &HFF)
                obj.System_KeyUp k, j
          End If
   Case WM_MOUSEWHEEL
          Case WM_MOUSEMOVE
                If (lFlags And mkfMouseMove) = mkfMouseMove Then
                      If GetAsyncKeyState(vbKeyLButton) Then i = 1
                      If GetAsyncKeyState(vbKeyRButton) Then i = 2
                      If GetAsyncKeyState(vbKeyMButton) Then i = 4
                      If GetAsyncKeyState(vbKeyShIft) Then j = 1
                      If GetAsyncKeyState(vbKeyControl) Then j = 2
                      If GetAsyncKeyState(vbKeyMenu) Then j = 4
                      obj.System_MouseMove i, j, CSng(em.lParamLow), CSng(em.lParamHigh)
                End If
          Case WM_LBUTTONDOWN, WM_RBUTTONDOWN, WM_MBUTTONDOWN
                If (lFlags And mkfMouseDown) = mkfMouseDown Then
                      If GetAsyncKeyState(vbKeyShIft) Then i = 1
                      If GetAsyncKeyState(vbKeyControl) Then i = 2
                      If GetAsyncKeyState(vbKeyMenu) Then i = 4
                      obj.System_MouseDown 2 ^ ((em.wMsg - 513) / 3), i, CSng(em.lParamLow), CSng(em.lParamHigh)
                End If
          Case WM_LBUTTONUP, WM_RBUTTONUP, WM_MBUTTONUP
                If (lFlags And mkfMouseUp) = mkfMouseUp Then
                      If GetAsyncKeyState(vbKeyShIft) Then i = 1
                      If GetAsyncKeyState(vbKeyControl) Then i = 2
                      If GetAsyncKeyState(vbKeyMenu) Then i = 4
                      obj.System_MouseUp 2 ^ ((em.wMsg - 514) / 3), i, CSng(em.lParamLow), CSng(em.lParamHigh)
                End If
          End Select
    Call CallNextHookEx(lHook, lCode, wParam, lParam)
End Function

Public Sub SetHook(frmOwner As Form, mkf As MouseKeyFlags)
   lHook = SetWindowsHookEx(WH_JOURNALRECORD, AddressOf fHookApp, 0, 0)
   Set obj = frmOwner
   lFlags = mkf
   fOnTop obj.hwnd, True
End Sub

Public Sub RemoveHook()
   UnhookWindowsHookEx lHook
```

```
  fOnTop obj.hwnd, False
  Set obj = Nothing
End Sub

Private Function fOnTop(hwnd As Long, bAlwaysOnTop As Boolean) As Boolean
  fOnTop = SetWindowPos(hwnd, -2 - bAlwaysOnTop, 0, 0, 0, 0, _
                                    SWP_NOREDRAW Or SWP_NOSIZE Or _
                                    SWP_NOMOVE)
End Function
```

IN FORM:

```
Private Declare Function GetForegroundWindow& Lib "user32" ()

Private Declare Function GetWindowThreadProcessId& Lib _
                                    "user32" (ByVal hwnd As Long, _
                                    lpdwProcessId As Long)

Private Declare Function GetKeyboardLayout& Lib "user32" _
                                    (ByVal dwLayout As Long)

Private Declare Function MapVirtualKeyEx Lib "user32" Alias _
                                    "MapVirtualKeyExA" _
                                    (ByVal uCode As Long, _
                                    ByVal uMapType As Long, _
                                    ByVal dwhkl As Long) _
                                    As Long

Private Sub Command1_Click()
   Unload Me
   End
End Sub

Private Sub Form_Load()
  SetHook Me, mkfMouseDown + mkfMouseUp + mkfMouseMove + mkfKeyDown + mkfKeyUp
  Text1 = "Log Mouse Aktivitaet :"
  Text2 = "Log Keyboard Aktivitaet :"
End Sub

Public Sub System_KeyDown(KeyCode As Integer, Shift As Integer)
  Dim s   As String

  s = "Tastatur-Code " & KeyCode
  s = s + fKeyCode(KeyCode)
  If Shift = vbShiftMask Then s = s & " + Shift "
  If Shift = vbCtrlMask Then s = s & " + Ctrl "
  If Shift = vbAltMask Then s = s & " + Alt "
  Text2 = Text2 & vbCrLf & s & " down"
End Sub

Public Sub System_KeyUp(KeyCode As Integer, Shift As Integer)
  Dim s   As String

  s = "Tastatur-Code " & KeyCode
  s = s + fKeyCode(KeyCode)
  If Shift = vbShiftMask Then s = s & " + Shift "
  If Shift = vbCtrlMask Then s = s & " + Ctrl "
  If Shift = vbAltMask Then s = s & " + Alt "
  Text2 = Text2 & vbCrLf & s & " up"
End Sub

Public Sub System_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
  Dim s    As String

  If Button = vbLeftButton Then s = "Linker Button "
  If Button = vbRightButton Then s = "Rechter Button "
  If Button = vbMiddleButton Then s = "Mittlerer Button "
  If Shift = vbShiftMask Then s = s & "+ Shift "
  If Shift = vbCtrlMask Then s = s & "+ Ctrl "
  If Shift = vbAltMask Then s = s & "+ Alt "
  Text1 = Text1 & vbCrLf & s & "Runter an Position (pixels): " & CStr(x) & " , " & (y)
End Sub

Public Sub System_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
  Dim s    As String

  If Button = vbLeftButton Then s = "Links Button "
```

```
  If Button = vbRightButton Then s = "Rechts Button "
  If Button = vbMiddleButton Then s = "Mittlerer Button "
  If Shift = vbShiftMask Then s = s & "+ Shift "
  If Shift = vbCtrlMask Then s = s & "+ Ctrl "
  If Shift = vbAltMask Then s = s & "+ Alt "
  Text1 = Text1 & vbCrLf & s & "Rauf auf Position (pixels): " & CStr(x) & " , " & CStr(y)
End Sub

Public Sub System_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
  Dim s     As String

  If Button = vbLeftButton Then s = "Links Button "
  If Button = vbRightButton Then s = "Rechts Button "
  If Button = vbMiddleButton Then s = "Mittlerer Button "
  If Shift = vbShiftMask Then s = s & "+ Shift "
  If Shift = vbCtrlMask Then s = s & "+ Ctrl "
  If Shift = vbAltMask Then s = s & "+ Alt "
  Label1 = "Mouse Information" & vbCrLf & "X = " & x & "  Y= " & y & vbCrLf
  If s <> "" Then Label1 = Label1 & "Information : " & vbCrLf & s & "pressed"
End Sub

Private Sub Form_Unload(Cancel As Integer)
   RemoveHook
End Sub

Private Function fKeyCode(k As Integer) As String
   Dim s    As String

   Select Case k
        Case vbKeyBack:     s = "BackSpace"
        Case vbKeyTab:      s = "Tab"
        Case vbKeyClear:    s = "Clear"
        Case vbKeyReturn:   s = "Enter"
        Case vbKeyShift:    s = "Shift"
        Case vbKeyControl:  s = "Ctrl"
        Case vbKeyMenu:     s = "Alt"
        Case vbKeyPause:    s = "Pause"
        Case vbKeyCapital:  s = "CapsLock"
        Case vbKeyEscape:   s = "ESC"
        Case vbKeySpace:    s = "SPACEBAR"
        Case vbKeyPageUp:   s = "PAGE UP"
        Case vbKeyPageDown: s = "PAGE DOWN"
        Case vbKeyEnd:      s = "END"
        Case vbKeyHome:     s = "HOME"
        Case vbKeyLeft:     s = "LEFT ARROW"
        Case vbKeyUp:       s = "UP ARROW"
        Case vbKeyRight:    s = "RIGHT ARROW"
        Case vbKeyDown:     s = "DOWN ARROW"
        Case vbKeySelect:   s = "SELECT"
        Case vbKeyPrint:    s = "PRINT SCREEN"
        Case vbKeyExecute:  s = "EXECUTE"
        Case vbKeySnapshot: s = "SNAPSHOT"
        Case vbKeyInsert:   s = "INS"
        Case vbKeyDelete:   s = "DEL"
        Case vbKeyHelp:     s = "HELP"
        Case vbKeyNumlock:  s = "NUM LOCK"
        Case vbKey0 To vbKey9: s = Chr$(k)
        Case vbKeyA To vbKeyZ: s = Chr$(MapVirtualKeyEx(k, 2,
GetKeyboardLayout(GetWindowThreadProcessId(GetForegroundWindow, 0))))
        Case vbKeyF1 To vbKeyF16: s = "F" & CStr(k - 111)
        Case vbKeyNumpad0 To vbKeyNumpad9: s = "Numpad " & CStr(k - 95)
        Case vbKeyMultiply: s = "Numpad {*}"
        Case vbKeyAdd: s = "Numpad {+}"
        Case vbKeySeparator: s = "Numpad {ENTER}"
        Case vbKeySubtract: s = "Numpad {-}"
        Case vbKeyDecimal: s = "Numpad {.}"
        Case vbKeyDivide: s = "Numpad {/}"
        Case Else
        s = Chr$(MapVirtualKeyEx(k, 2, GetKeyboardLayout(GetWindowThreadProcessId(GetForegroundWindow, 0))))
   End Select
   fKeyCode = "[" & s & " key]"
End Function
```

Tipp : 29.

IN FORM:

```
'Beispiel : Passwortzeicheneingabe in einer
'          Inputbox

Private Sub Command1_Click()
  Dim sRes              As String

  SetTimer hwnd, AP_INPUTB, 10, AddressOf sTimerProcess
  sRes = InputBox("Passwort eingeben :")
End Sub

Private Sub Command2_Click()
    End
End Sub
```

IN MODUL:

```
Option Explicit

Private Declare Function FindWindow Lib "user32" Alias _
                                    "FindWindowA" _
                                    (ByVal lpClassName As String, _
                                    ByVal lpWindowName As String) _
                                    As Long

Private Declare Function FindWindowEx Lib "user32" Alias _
                                    "FindWindowExA" (ByVal hWnd1 As Long, _
                                    ByVal hWnd2 As Long, _
                                    ByVal lpsz1 As String, _
                                    ByVal lpsz2 As String) _
                                    As Long

Public Declare Function SetTimer& Lib "user32" _
                                    (ByVal hwnd&, _
                                    ByVal nIDEvent&, _
                                    ByVal uElapse&, _
                                    ByVal lpTimerFunc&)

Private Declare Function KillTimer& Lib "user32" _
                                    (ByVal hwnd&, _
                                    ByVal nIDEvent&)

Private Declare Function SEndMessage Lib "user32" Alias _
                                    "SendMessageA" _
                                    (ByVal hwnd As Long, _
                                    ByVal wMsg As Long, _
                                    ByVal wParam As Long, _
                                    lParam As Any) _
                                    As Long

Const EM_PASSWORDCHAR = &HCC
Public Const AP_INPUTB As Long = &H5000&

Public Sub sTimerProcess(ByVal hwnd&, ByVal uMsg&, ByVal bvIdEvent&, ByVal dwTime&)
    Dim lRet            As Long

    lRet = FindWindowEx(FindWindow("#32770", App.Title), 0, "Edit", "")
    Call SendMessage(lRet, EM_PASSWORDCHAR, Asc("*"), 0)
    KillTimer hwnd, bvIdEvent
End Sub
```

Tipp : 30.

Mausposition ermitteln

```
Private Declare Function GetCursorPos Lib "user32" _
                                    (lpPoint As POINTAPI) _
                                    As Long

Private Declare Function SetCursorPos Lib "user32" (ByVal X As Long, _
                                    ByVal Y As Long) _
                                    As Long

Private Type POINTAPI
    X       As Long
    Y       As Long
```

```
End Type

Private pa          As POINTAPI
Private lRet        As Long

Public Function fGetMouseXPos() As Long
   lRet = GetCursorPos(pa)
   fGetMouseXPos = pa.X
End Function

Public Function fGetMouseYPos() As Long
   lRet = GetCursorPos(pa)
   fGetMouseYPos = pa.Y
End Function

Public Sub fSetMouseXPos(X As Long)
   lRet = GetCursorPos(pa)
   lRet = SetCursorPos(X, pa.Y)
End Sub

Public Sub fSetMouseYPos(Y As Long)
   lRet = GetCursorPos(pa)
   lRet = SetCursorPos(pa.X, Y)
End Sub

Public Sub fSetMousePos(X As Long, Y As Long)
   lRet = GetCursorPos(pa)
   lRet = SetCursorPos(pa.X, Y)
   lRet = GetCursorPos(pa)
   lRet = SetCursorPos(X, pa.Y)
End Sub

Private Sub Command1_Click()
   fSetMouseXPos 100
End Sub

Private Sub Command2_Click()
   fSetMouseYPos 100
End Sub

Private Sub Command3_Click()
   fSetMousePos 300, 200
End Sub

Private Sub Command4_Click()
   Unload Me
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Label1.Caption = fGetMouseXPos
Label2.Caption = fGetMouseYPos
End Sub
```

Tipp : 31.

```
'Beispiel : Ver-und Entschluesseln...

Public Function fDeCrypt(ByVal sString As String, _
                                    ByVal sSecurityValue As String) _
                                   As String
   For n = 1 To Len(sString)
          x = n Mod Len(sSecurityValue): If x = 0 Then x = Len(sSecurityValue)
          fDeCrypt = fDeCrypt & Chr(Asc(Mid(sSecurityValue, x, 1)) Xor Asc(Mid(sString, n, 1)))
   Next n
End Function

Private Sub Command1_Click()
   Text1 = fDeCrypt(Text1, Text2)
End Sub

Private Sub Command2_Click()
   End
End Sub
```

Tipp : 32.

```
'Beispiel : Tabelle einer Datenbank umbenennen
'Hinweis  : Erstellen Sie eine Referenz zur MS-DAO 3.51
'          Object Library
'Auf eine Fehlerbehandlung wurde in diesem Beispiel ver-
'zichtet. Stellen Sie also sicher das sich die Tabelle die
'umbenannt werden soll in der Datenbank befindet...
'Rueckgabe: True wenn erfolgreich, sonst False
'sDatabaseName: ist der komplette sPath
'sOldTableName: ist die Tabelle die umbenannt werden soll
'sNewTableName: ist der Neu vergebene Tabellenname...

Dim sPath As String

Private Function fRenameTable(sDatabaseName As String, ByVal sOldTableName As String, _
                                                                ByVal sNewTableName As String) _
                                                               As Boolean
   On Error GoTo ErrHandle:
   Dim db                  As DAO.Database
   Dim td                  As DAO.TableDef

   Set db = Workspaces(0).OpEndatabase(sDatabaseName)
   On Error GoTo Fehler
   Set td = db.TableDefs(sOldTableName)
   td.Name = sNewTableName
   db.TableDefs.Refresh
   db.Close
   fRenameTable = True
   Exit Function
ErrHandle:
   If Not db Is Nothing Then db.Close
   Set td = Nothing
End Function

Private Sub Command1_Click()
   fRenameTable sPath, Text2, Text3
End Sub

Private Sub Command2_Click()
   Unload Me
   End
End Sub

Private Sub Form_Load()
  sPath = App.Path & "\" & "test.mdb"
  Text1 = sPath
  Text2 = "Test2"
  Text3 = "Test2_Test2_Neu"
End Sub
```

Tipp : 33.

```
'Beispiel : Grafik in die Zwischenablage einfuegen und wieder
'abrufen...

Option Explicit

Dim sgx1          As Single
Dim sgy1          As Single
Dim sgx2          As Single
Dim sgy2          As Single
Dim sgx1a         As Single
Dim sgy1a         As Single
Dim sgx1b         As Single
Dim sgx1b         As Single

Private Sub Command1_Click(Index As Integer)
   Select Case Index
   Case 0
          Form1.Cls
   Case 1
          Unload Me
          End
   End Select
End Sub
```

```
Private Sub Form_Load()
   Picture1.CurrentY = 200 '550
   Picture1.CurrentX = 200
   Picture1.Print "Markieren Sie hier einen Bereich"
   Picture1.CurrentX = 200
   Picture1.Print "indem Sie die linke Maustaste gedrueckt"
   Picture1.CurrentX = 200
   Picture1.Print "halten und einen Rahmen ziehen..."
   Picture1.CurrentX = 200
   Picture1.Print "F R E E   D O W N L O A D"
   Picture1.CurrentX = 200
   Picture1.Print "by"
   Picture1.CurrentX = 200
   Picture1.Print "http://www.Visual-Basic5.de"
   Picture1.CurrentX = 200
   Picture1.Print "All Rights reserved by"
   Picture1.CurrentX = 200
   Picture1.Print "Heinz Prelle Germany"
   Picture1.CurrentX = 200
   Picture1.Print "outa.space@t-online.de"
   Clipboard.Clear
   Form1.Show
   DoEvents
End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
      If Button = 1 Then
            If sgx1 <> sgx2 Then
                  Picture1.Line (sgx1, sgy1)-(sgx2, sgy2), , B
                  sgx1 = 0
                  sgy1 = 0
                  sgx2 = 0
                  sgy2 = 0
            End If
            sgx1a = X
            sgy1a = Y
            sgx1b = X
            sgx1b = Y
      End If
End Sub

Private Sub Form_MouseMove(Button As Integer, ShIft As Integer, X As Single, Y As Single)
   If Button = 1 Then
            Form1.DrawMode = vbInvert
            Form1.DrawStyle = vbDot
            If sgx1b > 0 Then Form1.Line (sgx1a, sgy1a)-(sgx1b, sgx1b), , B
                  Form1.Line (sgx1a, sgy1a)-(X, Y), , B
                  sgx1b = X
                  sgx1b = Y
   End If
End Sub

Private Sub Form_MouseUp(Button As Integer, ShIft As Integer, X As Single, Y As Single)
   On Error GoTo Fehler
   If Button = 1 Then
            If sgx1a <> sgx1b Then
                  Form1.Line (sgx1a, sgy1a)-(sgx1b, sgx1b), , B
                  PaintPicture Clipboard.GetData, sgx1a, sgy1a, sgx1b - sgx1a, sgx1b - sgy1a
            Else
                  PaintPicture Clipboard.GetData, X, Y
            End If
   End If
   Exit Sub

ErrHandle:
   If Err = 481 Then MsgBox _
   "Fehler. Sie haben keinen Ausschnitt in die Zwischenablage kopiert...", 16, "Anwenderfehler..."
End Sub

Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If sgx2 > 0 Then
            Picture1.Line (sgx1, sgy1)-(sgx2, sgy2), , B
            sgx2 = X
            sgy2 = Y
   End If
   sgx1 = X
```

```
    sgy1 = Y
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 Then
            Picture1.DrawMode = vbInvert
            Picture1.DrawStyle = vbDot
            If sgx2 > 0 Then Picture1.Line (sgx1, sgy1)-(sgx2, sgy2), , B
            Picture1.Line (sgx1, sgy1)-(X, Y), , B
            sgx2 = X
            sgy2 = Y
    End If
End Sub

Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
      On Error Resume Next
      If X <> sgx1 Then
            Picture2.Cls
            Picture2.ScaleMode = vbTwips
            If sgx1 > sgx2 Then sgx2 = sgx1: sgx1 = X
            If sgy1 > sgy2 Then sgy2 = sgy1: sgy1 = Y
            Picture2.Width = (sgx2 - sgx1) + 60
            Picture2.Height = (sgy2 - sgy1) + 60
            Picture1.DrawMode = vbInvert
            Picture1.DrawStyle = vbDot
            Picture1.Line (sgx1, sgy1)-(sgx2, sgy2), , B
             Picture2.PaintPicture Picture1.Image, 0, 0, (sgx2 - sgx1), (sgy2 - sgy1), sgx1, sgy1, (sgx2 - sgx1), _
                                   (sgy2 - sgy1)
            Picture1.Line (sgx1, sgy1)-(sgx2, sgy2), , B
            Clipboard.Clear
            Clipboard.SetData Picture2.Image
      End If
End Sub
```

Tipp : 34.

```
'Beispiel : Test ob eine Soundkarte installiert ist
'Rueckgabe: True wenn ja, sonst False...

Private Declare Function waveOutGetNumDevs Lib "winmm.dll" () As Long

Public Function SoundCardInstalled() As Boolean
    SoundCardInstalled = waveOutGetNumDevs > 0
End Function

Private Sub Command1_Click()
    Frame1.Caption = "Soundkarte installiert : " & SoundCardInstalled
End Sub

Private Sub Command2_Click()
    Unload Me
End Sub
```

Tipp : 35.

```
'Beispiel : Daten an Excel senden...
'Hinweis  : Stellen Sie eine Referenz zur MS-Excel 8.0
'Object Library her.
Dim objExcel As Excel.Application

Private Sub Command1_Click()
    objExcel.Workbooks.Add
    objExcel.Range(Text1.Text).Value = Text2.Text
    objExcel.Visible = True
End Sub

Private Sub Command2_Click()
    End
End Sub

Private Sub Form_Load()
    Set objExcel = CreateObject("Excel.Application")
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    objExcel.Quit
End Sub
```

Tipp : 36.

```
'Beispiel : Word aufrufen
'Hinweis  : Stellen Sie eine Referenz zur MS-Word 8.0
'Object Library her.
'Hinweis : Auf eine Fehlerbehandlung wurde verzichtet.
Dim objWord          As Word.Application

Private Sub Command1_Click()
    objWord.Documents.Add
    '
    objWord.WindowState = wdWindowStateMaximize
    objWord.Visible = True
End Sub

Private Sub Command2_Click()
    objWord.Quit
    Set objWord = Nothing
    End
End Sub

Private Sub Form_Load()
    Set objWord = CreateObject("Word.Application")
End Sub

Private Sub Form_Unload(Cancel As Integer)
    objWord.Quit
    Set objWord = Nothing
End Sub
```

Tipp : 37.

```
Option Explicit
'Beispiel : Informationen einer .wav - sFile auslesen...

Private Const coRIFFID = 1179011410
Private Const coRIFFWAV = 1163280727
Private Const coRIFFFORMAT = 544501094

'Typischer Haeder einer .wav-sFile

Private Type WaveInformation
    wiRiffFormat                As Long
    lHeight                     As Long
    lId                         As Long
    wiFormat                    As Long
    wiWaveFormat                As Integer
    wiChannels                  As Integer
    wiSmpPerSecond              As Long
    wiBytesPerSecond            As Long
    wiBlockQuote                As Integer
End Type

Private Function fGetWaveInfos(ByVal sFileName As String, ByRef wi As WaveInformation) As Boolean
    Dim iFileNumber As Integer

    iFileNumber = FreeFile
    On Error GoTo ErrHandle
    Open sFileName For Binary Access Read As #iFileNumber
            On Error GoTo FileError
            Get #iFileNumber, , wi
    Close #iFileNumber

    On Error GoTo Allgemeiner_Fehler

    If wi.wiRiffFormat = coRIFFID And wi.lId = coRIFFWAV And wi.wiFormat = coRIFFFORMAT _
                                    Then
            fGetWaveInfos = True
    Else
            fGetWaveInfos = False
```

```
    End If
    Exit Function
FileError:
    Close #iFileNumber
ErrHandle:
    fGetWaveInfos = False
End Function

Private Sub Command1_Click()
   Dim wi           As WaveInformation
   Dim sPath        As String

   sPath = App.Path + "\test.wav"
   If fGetWaveInfos(sPath, wi) Then
            With wi
                     Label1(0).Caption = wi.wiRiffFormat
                     Label1(1).Caption = wi.lHeight
                     Label1(2).Caption = wi.lId
                     Label1(3).Caption = wi.wiFormat
                     Label1(4).Caption = wi.wiWaveFormat
                     Label1(5).Caption = wi.wiChannels '0 = Mono 1 = Stereo
                     Label1(6).Caption = wi.wiSmpPerSecond
                     Label1(7).Caption = wi.wiBytesPerSecond '11025kHz usw.
                     Label1(8).Caption = wi.wiBlockQuote
            End With
   End If
End Sub

Private Sub Command2_Click()
    End
End Sub
```

Tipp : 38.

```
Combobox ferngesteuert öffnen und schliessen

Private Declare Function SendMessage Lib "user32" Alias _
                                    "SendMessageA" _
                                    (ByVal hwnd As Long, _
                                    ByVal wMsg As Long, _
                                    ByVal wParam As Long, _
                                    lParam As Any) _
                                    As Long

Private Const CB_SHOWDROPDOWN = &H14F

Private Sub Command1_Click()
    lRet = SendMessage(Combo1.hwnd, CB_SHOWDROPDOWN, True, 0)
End Sub

Private Sub Command3_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Combo1.AddItem "www.Visual-Basic5.de"
    Combo1.AddItem "www.vb-5.de"
    Combo1.AddItem "outa.space@t-online.de"
    Combo1.ListIndex = 0
End Sub
```

Tipp : 39.

```
'Beispiel : Test ob Programm schon geladen wurde...

Public Sub fSoftwareCheck(obj As Form)
    Dim sTitle          As String

    If App.PrevInstance Then
            sTitle$ = App.Title
            MsgBox "Das Programm wurde schon gestartet..."
            App.Title = ""
            obj.Caption = ""
            AppActivate sTitle$
```

```
    SendKeys "%{ENTER}", True
    End
    End If
End Sub

Private Sub Command1_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Call fSoftwareCheck(Form1)
End Sub
```

Tipp : 40.

```
'Beispiel : Suchen und Ersetzen...

Function fReplaceWith(sSearchingString As String, sSearchWath As String, sReplaceWith As String)
    Dim sTmp  As String, iFound  As Integer

    iFound  = InStr(sSearchintText, sSearchWath): sTmp = sSearchintText
    If iFound  <> 0 Then
            sTmp = ""
            If iFound  > 1 Then sTmp = Left(sSearchintText, iFound  - 1)
            sTmp = sTmp + sReplaceWith
                    If iFound  + Len(sSearchWath) - 1 < Len(sSearchintText) Then sTmp =  sTmp + _
                    Right$(sSearchintText, _
                    Len(sSearchintText) - iFound  - Len(sSearchWath) + 1)
            End If
    fReplaceWith = strsSearchintText
End Function

Private Sub Command1_Click()
            Text1 = fReplaceWith(Text1, Text2, Text3)
End Sub

Private Sub Command2_Click()
            Unload Me
End Sub
```

Tipp : 41.

```
'Beispiel : Form zentrieren....

Public Sub sCenterFormToScreen (obj As Form)
    obj.Move (Screen.Width - obj.Width) \ 2,  (Screen.Height - obj.Height) \ 2
    Load obj
    obj.Show
End Sub

Private Sub Command1_Click()
    sCenterFormToScreen  Me
End Sub

Private Sub Command2_Click()
    Unload Me
End Sub
```

Tipp : 42.

```
'Beispiel : Freien Festplattenplatz in KB ermitteln...
'Hinweis  : Auf eine Fehlerbehandlung wurde verzichtet.

Private Declare Function GetDiskFreeSpace Lib "kernel32" _
                                    Alias "GetDiskFreeSpaceA" _
                                    (ByVal lpRootPathName As String, _
                                    lpSectorsPerCluster As Long, _
                                    lpBytesPerSector As Long, _
                                    lpNumberOfFreeClusters As Long, _
                                    lpTtoalNumberOfClusters As Long) _
                                    As Long

Public Function fGetFreeDiskSpace (strRootPathName As String) As Long
```

```
    Dim lngSectpCluster         As Long
    Dim lngBytespSector         As Long
    Dim lngFreeCluster          As Long
    Dim lngTotalCluster         As Long

    GetDiskFreeSpace strRootPathName, lngSectpCluster, lngBytespSector, lngFreeCluster, lngTotalCluster
   fGetFreeDiskSpace  = lngFreeCluster * lngSectpCluster * lngBytespSector / 1024
End Function

Private Sub Command1_Click()
    Dim sPath As String
    sPath = "c:\"
    On Error Resume Next
    Frame1.Caption = fGetFreeDiskSpace (sPath) & " Kb auf sDrive " & sPath
End Sub

Private Sub Command2_Click()
    Unload Me
End Sub
```

Tipp : 43.

```
'Beispiel : Gesamten Inhalt einer HD anzeigen...
'HINWEIS  :
'List1.Visible = False
'Dir1.Visible = False
'File1.Visible = False

Private Sub Command1_Click()
    'Clear ListBoxen
    vBeginingTime  = Time
    List1.Clear
    List2.Clear
    'sPath auf Root-Verzeichnis setzen
    Dir1.Path = Left$(Drive1.Drive, 2) + "\"
    File1.Path = Dir1.Path  'sFilesPath setzen
    'sFileen im Root-Verzeichnis aufnehmen
    For n  = 0 To File1.ListCount - 1
            If Right$(Dir1.Path, 1) <> "\" Then
                    List2.AddItem Dir1.Path + "\" + File1.List(n )
            Else
                    List2.AddItem Dir1.Path + File1.List(n )
            End If
    Next
    'Unterverzeichnisse aufnehmen
    For n  = 0 To Dir1.ListCount - 1
            List1.AddItem Dir1.List(n )
    Next
    'wenn keine Unterverzeichnisse vorhanden
    If n  = 0 Then GoTo AustrittsPunkt
    lStart&  = 0  'Start mit erstem Verzeichnis in der Liste
EntryPoint:
    DoEvents 'System freigeben...
    Dir1.Path = List1.List(lStart& )
    File1.Path = Dir1.Path

    'Alle gefundenen Unterverzeichnisse aufnehmen
    For n  = 0 To Dir1.ListCount - 1
            List1.AddItem Dir1.List(n )
    Next

    'Alle gefundenen sFileen aufnehmen
    For n  = 0 To File1.ListCount - 1
            List2.AddItem Dir1.Path + "\" + File1.List(n )
    Next

    lStart&  = lStart&  + 1
    If lStart&  < List1.ListCount Then GoTo EntryPoint
AustrittsPunkt:
    Label1.Caption = "Gesamt:" + Str$(List2.ListCount)
    vEndTime = Time
    Label2.Caption = "Vorgang gestartet um : " & _
    Format(vBeginingTime , "hh:mm:ss") & ". Beendet nach : " & Format(vBeginingTime  - vEndTime, "hh:mm:ss")
End Sub

Private Sub Command2_Click()
```

```
    Unload Me
End Sub
```

Tipp : 44.

```
'Beispiel : Druckt eine Grafik so gross wie moeglich...
'           PassEnd auf die Druckermoeglichkeiten.

Public Function fPrintImage(objPrinter As Printer, pb As Picture) As Boolean

    Const vbHiMetric As Integer = 8

    Dim dblImgCoordinates           As Double
    Dim dblPrinterWidth             As Double
    Dim dblPrinterHeight            As Double
    Dim dblPrinterCoordinates       As Double
    Dim dblImgWidth                 As Double
    Dim dblImgHeight                As Double

    On Error GoTo ErrHandle:
    'Seitenorientierung setzen [Hoch oder Querformat]
    If pb.Height >= pb.Width Then
            objPrinter.Orientation = vbPRORPortrait
    Else
            objPrinter.Orientation = vbPRORLandscape
    End If
    'Berechnung der Geraeteunabhaengigkeit.Weiten- und
    'Hoehenverhaeltnis der Grafik.
    dblImgCoordinates = pb.Width / pb.Height
    'Berechnung des Druckbereiches in HiMetric
    dblPrinterWidth = objPrinter.ScaleX(objPrinter.ScaleWidth, _
    objPrinter.ScaleMode, vbHiMetric)
    dblPrinterHeight = objPrinter.ScaleY(objPrinter.ScaleHeight, _
    objPrinter.ScaleMode, vbHiMetric)
    'Berechnung der Geraeteunabhaengigkeit. Weiten- und
    'Hoehenverhaelltnis des Druckers.
    dblPrinterCoordinates = dblPrinterWidth / dblPrinterHeight
    'Ausgabescalierung des Druckbereiches.
    If dblImgCoordinates >= dblPrinterCoordinates Then
            'Bild scalieren auf den vollen Druckbereich.(Weite)
            dblImgWidth = objPrinter.ScaleX(dblPrinterWidth, vbHiMetric, objPrinter.ScaleMode)
            dblImgHeight = objPrinter.ScaleY(dblPrinterWidth / dblImgCoordinates, vbHiMetric, objPrinter.ScaleMode)
    Else
            'Bild scalieren auf den vollen Druckbereich.(Hoehe)
            dblImgHeight = objPrinter.ScaleY(dblPrinterHeight, _
            vbHiMetric, objPrinter.ScaleMode)
            dblImgWidth = objPrinter.ScaleX(dblPrinterHeight * dblImgCoordinates, vbHiMetric, objPrinter.ScaleMode)
    End If
    'Bild drucken mit der PaintPicture Methode.
    objPrinter.PaintPicture pb, 0, 0, dblImgWidth, _
    dblImgHeight
    fPrintImage = True
    objPrinter.EndDoc '
    Exit Function
ErrHandle:
    fPrintImage = False
End Function

Private Sub Command1_Click()
    fPrintImage Printer, Picture1
End Sub

Private Sub Command2_Click()
    Unload Me
End Sub
```

Tipp : 45.

```
'Beispiel : URL aus VB starten...

Private Declare Function ShellExecute Lib "shell32.dll" _
                                  Alias "ShellExecuteA" _
                                  (ByVal hwnd As Long, _
                                  ByVal lpOperation As String, _
```

```
                                ByVal lpFile As String, _
                                ByVal lpParameters As String, _
                                ByVal lpDirectory As String, _
                                ByVal nShowCmd As Long) _
                                As Long

Private Declare Function GetDesktopWindow Lib "user32" () _
                                As Long

Const SW_SHOWNORMAL = 1
Dim URL             As String

Public Function fStartUrl(sUrl As String) As Long
   Dim ScrHDC                   As Long

   ScrHDC = GetDesktopWindow()
   fStartUrl = ShellExecute(ScrHDC, "Open", sUrl, vbNullString, "C:\", SW_SHOWNORMAL)
End Function

Private Sub Command1_Click()
   Call fStartUrl(URL)
End Sub

Private Sub Command2_Click()
   End
End Sub

Private Sub Form_Load()
   URL = "http://www.Visual-Basic5.de"
   Command1.Caption = URL
End Sub
```

Tipp : 46.

```
'Beispiel : sFile Oeffnen.../Speichern...-Dialog
'           ohne ActiveX - Komponente.

Option Explicit

Private Type OPENFILENAME
   lStructSize          As Long
   hwndOwner            As Long
   hInstance            As Long
   lpsFilter            As String
   lpstrCustomFilter    As String
   nMaxCustFilter       As Long
   nFilterIndex         As Long
   lpstrFile            As String
   nMaxFile             As Long
   lpstrFileTitle       As String
   nMaxFileTitle        As Long
   lpstrInitialDir      As String
   lpstrTitle           As String
   flags                As Long
   nFileOffset          As Integer
   nFileExtension       As Integer
   lpstrDefExt          As String
   lCustData            As Long
   lpfnHook             As Long
   lpTemplateName       As String
End Type

Private Const OFN_READONLY = &H1
Private Const OFN_OVERWRITEPROMPT = &H2
Private Const OFN_HIDEREADONLY = &H4
Private Const OFN_NOCHANGEDIR = &H8
Private Const OFN_SHOWHELP = &H10
Private Const OFN_ENABLEHOOK = &H20
Private Const OFN_ENABLETEMPLATE = &H40
Private Const OFN_ENABLETEMPLATEHANDLE = &H80
Private Const OFN_NOVALIDATE = &H100
Private Const OFN_ALLOWMULTISELECT = &H200
Private Const OFN_EXTENSIONDIFFERENT = &H400
Private Const OFN_PATHMUSTEXIST = &H800
Private Const OFN_FILEMUSTEXIST = &H1000
Private Const OFN_CREATEPROMPT = &H2000
```

```
Private Const OFN_SHAREAWARE = &H4000
Private Const OFN_NOREADONLYRETURN = &H8000
Private Const OFN_NOTESTFILECREATE = &H10000
Private Const OFN_NONETWORKBUTTON = &H20000
Private Const OFN_NOLONGNAMES = &H40000
Private Const OFN_EXPLORER = &H80000
Private Const OFN_NODEREFERENCELINKS = &H100000
Private Const OFN_LONGNAMES = &H200000

Private Const OFN_SHAREFALLTHROUGH = 2
Private Const OFN_SHARENOWARN = 1
Private Const OFN_SHAREWARN = 0

Private Declare Function GetSaveFileName Lib "comdlg32.dll" Alias _
                                "GetSaveFileNameA" _
                                (pOpenfilename As OPENFILENAME) _
                                As Long

Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias _
                                "GetOpenFileNameA" _
                                (pOpenfilename As OPENFILENAME) _
                                As Long


Function fOpenDlg(frm As Form, sFilter As String, sTitle As String, sInitDir As String) As String
    Dim of                      As OPENFILENAME
    Dim lRet                    As Long

    of.lStructSize = Len(of)
    of.hwndOwner = frm.hWnd
    of.hInstance = App.hInstance

    If Right$(sFilter, 1) <> "|" Then sFilter = sFilter + "|"
    For lRet = 1 To Len(sFilter)
            If Mid$(sFilter, lRet, 1) = "|" Then Mid$(sFilter, lRet, 1) = Chr$(0)
    Next

    Me.Caption = sFilter

        of.lpsFilter = sFilter
        of.lpstrFile = Space$(254)
        of.nMaxFile = 255
        of.lpstrFileTitle = Space$(254)
        of.nMaxFileTitle = 255
        of.lpstrInitialDir = sInitDir
        of.lpstrTitle = sTitle
        of.flags = OFN_HIDEREADONLY Or OFN_FILEMUSTEXIST
        lRet = GetOpenFileName(of)

        If (lRet) Then
            fOpenDlg = Trim$(of.lpstrFile)
        Else
            fOpenDlg = ""
        End If
End Function

Function fSaveDlg(frm As Form, sFilter As String, sTitle As String, sInitDir As String) As String
        Dim of          As OPENFILENAME
        Dim lRet                    As Long

        of.lStructSize = Len(of)
        of.hwndOwner = frm.hWnd
        of.hInstance = App.hInstance
        If Right$(sFilter, 1) <> "|" Then sFilter = sFilter + "|"

        For lRet = 1 To Len(sFilter)
            If Mid$(sFilter, lRet, 1) = "|" Then Mid$(sFilter, lRet, 1) = Chr$(0)
        Next

        of.lpsFilter = sFilter
        of.lpstrFile = Space$(254)
        of.nMaxFile = 255
        of.lpstrFileTitle = Space$(254)
        of.nMaxFileTitle = 255
        of.lpstrInitialDir = sInitDir
        of.lpstrTitle = sTitle
        of.flags = OFN_HIDEREADONLY Or OFN_OVERWRITEPROMPT Or OFN_CREATEPROMPT
```

```
    lRet = GetSaveFileName(of)
    If (lRet) Then
        fSaveDlg = Trim$(of.lpstrFile)
    Else
        fSaveDlg = ""
    End If
End Function

Private Sub Command1_Click()
   fOpenDlg frm, "Text sFileen (*.txt)|*.txt|Alle sFileen (*.*)|*.*", "Öffnen...", "c:\"
End Sub

Private Sub Command2_Click()
   fSaveDlg frm, "Text sFileen (*.txt)|*.txt|Alle sFileen (*.*)|*.*", "Speichern...", "c:\"
End Sub

Private Sub Command3_Click()
   End
End Sub
```

Tipp : 47.

```
'Beispiel : Bestimmte Datei suchen, wenn vorhanden
'          Meldung ausgeben

Public Function fFileExists(sPath$) As Integer
   Dim x
   x = FreeFile
   On Error Resume Next
   Open sPath$ For Input As x
        If Err = 0 Then
             fFileExists = True
        Else
             fFileExists = False
             Frame2.Caption = "sFile nicht gefunden : > " & fFileExists
             Exit Function
        End If
   Close x
End Function

Public Function fSearchFile(ByVal sPath As String, ByVal sFile As String) As String
   Dim sDirName              As String
   Dim sLastDir              As String

   If sFile = "" Then Exit Function

   If Right(sPath, 1) <> "\" Then sPath = sPath & "\"
   sDirName = Dir(sPath & "*.*", vbDirectory)
   Do While Not fFileExists(sPath & sFile)
        If sDirName = "" Then Exit Do
        DoEvents
             If sDirName <> "." And sDirName <> ".." Then
                  If (GetAttr(sPath & sDirName) And vbDirectory) = vbDirectory Then
                       sLastDir = sDirName
                       sDirName = fSearchFile(sPath & sDirName & "\", sFile)
                            If sDirName <> "" Then
                                 sPath = sDirName
                                 Exit Do
                            End If
                            sDirName = Dir(sPath, vbDirectory)
                       Do Until sDirName = sLastDir Or sDirName = ""
                       sDirName = Dir
                       Loop
                  If sDirName = "" Then Exit Do
             End If
             End If
             sDirName = Dir
        Loop
   If fFileExists(sPath & sFile) Then fSearchFile = sPath
End Function

Private Sub Command1_Click()
   Dim sFileName   As String

   sFileName = Text1
   MsgBox fSearchFile("c:\", sFileName) & sFileName & " vorhanden.", vbOKOnly + vbInformation, "Meldung", 0, 0
```

```
End Sub

Private Sub Command2_Click()
   End
End Sub
```

Tipp : 48.

```
'Beispiel: Verzeichniswechsel
Option Explicit

Private sPath         As String

Private Sub Command1_Click()
   Dim sPath         As String
   sPath = fChangeFolder(Me, "Wählen Sie ein Verzeichnis :", Text1.Text)
   'Sub verlassen wenn Anwender <Abbrechen> waehlt
   If Len(sPath) = 0 Then Exit Sub
   Text1.Text = sPath
End Sub

Private Sub Command2_Click()
   End
End Sub

Private Sub Form_Load()
   Text1.Text = CurDir
End Sub
```

IN MODUL:

```
Option Explicit

Private Const WMUSER = &H400
Private Const CINIT = 1
Private Const SELECTEDCHANGED = 2
Private Const SETSTATUSTEXT = (WMUSER + 100)
Private Const SETCONNTENT = (WMUSER + 102)

Private Const C_STATUSTEXT = &H4&
Private Const C_RETURNONLYSYSDIRS = 1
Private Const C_DONTGOBD = 2
Private Const C_MAXPATH = 260

Private Declare Function SHGetPathFromIDList Lib "shell32" _
                                (ByVal pidList As Long, _
                                ByVal lpBuffer As String) _
                                As Long

Private Declare Function lstrcat Lib "kernel32" Alias _
                                "lstrcatA" (ByVal lpString1 As String, _
                                ByVal lpString2 As String) _
                                As Long

Private Declare Function SendMessage Lib "user32" Alias _
                                "SendMessageA" _
                                (ByVal hWnd As Long, _
                                ByVal wMsg As Long, _
                                ByVal wParam As Long, _
                                ByVal lParam As String) _
                                As Long

Private Declare Function SHBrowseForFolder Lib "shell32" _
                                (lpbi As ChangeInf) _
                                As Long

Private Type ChangeInf
  lOwnerhwnd                    As Long
  lRootId                       As Long
  lDispName                     As Long
  lTitle                        As Long
  flags                         As Long
  lCallBack                     As Long
  lParam                        As Long
  lImg                          As Long
End Type
```

```
Private sCurrDirAs String

Public Function fChangeFolder(obj As Form, sTitle As String, sRootDir As String) As String
   Dim strzsTitle              As String
   Dim sBuffer                 As String
   Dim lIdList                 As Long
   Dim ci                      As ChangeInf

   sCurrDir= sRootDir & vbNullChar
   strzsTitle = sTitle
      With ci
         .lOwnerhwnd = obj.hWnd
         .lTitle = lstrcat(strzsTitle, "")
         .flags = C_RETURNONLYSYSDIRS + C_DONTGOBD + C_STATUSTEXT
         .lCallBack = AdressOFFunction(AddressOf fCallback)  'get address of function.
      End With
   lIdList = SHBrowseForFolder(ci)
      If (lIdList) Then
         sBuffer = Space(C_MAXPATH)
         SHGetPathFromIDList lIdList, sBuffer
         sBuffer = Left(sBuffer, InStr(sBuffer, vbNullChar) - 1)
         fChangeFolder = sBuffer
      Else
         fChangeFolder = ""
      End If
End Function

Private Function fCallback(ByVal hWnd As Long, ByVal lMessage As Long, ByVal x As Long, ByVal lData As Long) _
                           As Long
Dim lIdList          As Long
Dim lRet             As Long
Dim sBuffer          As String

   On Error Resume Next
   Select Case lMessage
      Case CINIT
         Call SendMessage(hWnd, SETCONNTENT, 1, strAktuellesVerzeichnis)
      Case SELECTEDCHANGED
         sBuffer = Space(C_MAXPATH)
         lRet = SHGetPathFromIDList(x, sBuffer)
         If lRet = 1 Then
            Call SendMessage(hWnd, SETSTATUSTEXT, 0, sBuffer)
         End If
   End Select
   fCallback = 0
End Function

Private Function AdressOFFunction(lngAdd As Long) As Long
   AdressOFFunction = lngAdd
End Function
```

Tipp : 49.

```
'Beispiel : Verwendung ProgressBar.
'HINWEIS:
'Verwendet Microsoft Windows Common Controls 5.0 (SP 2)

Private Sub Command1_Click()
   End
End Sub

Private Sub Form_Load()
   Me.Show
   Timer1.Interval = 1000
   Timer1.Enabled = True
End Sub

Private Sub Timer1_Timer()
   vEndTime = 4   'Laufzeit der ProgressBar setzen
   ProgressBar1.Max = ((vEndTime + 1) * Timer1.Interval)
   Start = Timer   'Startzeit setzen.

   Do While Timer < Start + vEndTime
      ProgressBar1.Value = ProgressBar1.Value + 2
      DoEvents    'System frei lassen
```

```
    Loop

    Ende = Timer  'Endzeit setzen.
    vRunning = Ende - Start  'Gesamtzeit berechnen.
    MsgBox "Angehalten nach :" & vRunning & " Sekunden."
    Timer1.Enabled = False
End Sub
```

Tipp : 50.

```
'Beispiel : MouseMove

Private Declare Function ReleaseCapture Lib "user32" () As Long

Private Declare Function SendMessage Lib "user32" Alias _
                                    "SendMessageA" (ByVal hwnd As Long, _
                                    ByVal wMsg As Long, _
                                    ByVal wParam As Integer, _
                                    ByVal lParam As Long) _
                                    As Long


Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ReleaseCapture
    lRet& = SendMessage(Me.hwnd, &H112, &HF012, 0)
End Sub

Private Sub Command2_Click()
    End
End Sub
```