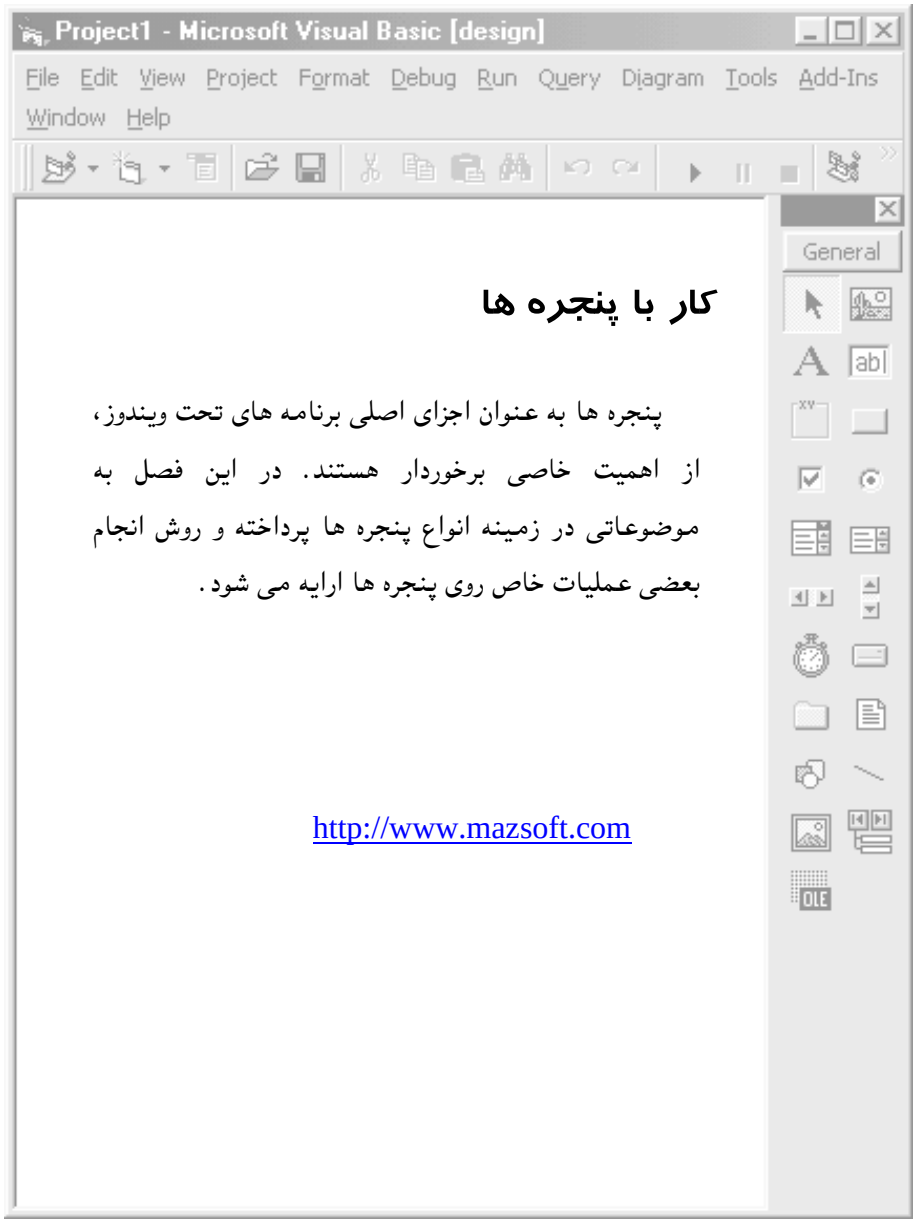
Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins
Window Help
General
کار با پنجره ها
پنجره ها به عنوان اجزای اصلی برنامه های تحت ویندوز، از اهمیت خاصی برخوردار هستند. در این فصل به موضوعاتی در زمینه انواع پنجره ها پرداخته و روش انجام بعضی عملیات خاص روی پنجره ها ارایه می شود.
http://www.mazsoft.com

تعیین وضعیت عادی یک پنجره مینیمایز شده

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
110620.TXT	۲۱ ژانویه ۱۹۹۵	3	3.1

در ویژوال بیسیک این امکان وجود ندارد که از طریق برنامه نویسی (بدون استفاده از تابع GetWindowPlacement) دریابیم یک پنجره که به حالت یک آیکون مینیمایز شده است پس از بازگشت به حال عادی به حالت معمولی در خواهد آمد یا ماکزیمایز خواهد شد. در این بخش خواهیم دید که چگونه از یک برنامه ویژوال بیسیک تابع فوق را فرابخوانیم.

برنامه زیر نشان می دهد که چگونه می توان دریافت که یک پنجره مینیمایز شده، پس از بازگشت به حال عادی ماکزیمایز خواهد شد یا وضعیت معمولی دارد. این برنامه به دو Form و یک Label و یک Command Button روی Form1 نیاز دارد.

۱- یک پروژه جدید در ویژوال بیسیک ایجاد کنید. Form1 بطور پیش فرض ایجاد می شود.

۲- از منوی File گزینه New Form را انتخاب کنید تا Form2 ایجاد شود.

۳- از منوی File گزینه New Module را انتخاب کنید تا یک ماژول جدید ایجاد شود و در بخش declarations آن لیست زیر را وارد کنید:

```
Type RECT
    left As Integer
    top As Integer
    right As Integer
    bottom As Integer
End Type

Type POINTAPI
    x As Integer
    y As Integer
End Type

Type WINDOWPLACEMENT
    length As Integer
    flags As Integer
    showCmd As Integer
    ptMinPosition As POINTAPI
    ptMaxPosition As POINTAPI
```

```
    rcNormalPosition As RECT
End Type

Global Const WPF_RESTORETOMAXIMIZED = &H0002

' Enter the following Declare statement as one, single line:
Declare Function GetWindowPlacement Lib "User" _
      (ByVal hWnd As Integer, lpwndpl As WINDOWPLACEMENT) As Integer

Function is_max (hWnd As Integer) As Integer
   Dim wp As WINDOWPLACEMENT
   Dim rtn As Integer
   wp.length = Len(wp) ' Initialize size
   rtn = GetWindowPlacement(hWnd, wp)
   If wp.flags = WPF_RESTORETOMAXIMIZED Then
       is_max = True
   Else
       is_max = False
   End If
End Function
```

☜ توجه : پارامتر wp.length باید دارای مقدار اولیه باشد، در غیر اینصورت تابع GetWindowPlacement خطا بر می گرداند.

۴- یک Label و یک Command Button به Form1 اضافه کنید.

۵- لیست زیر را به واقعه Click از Command1 اضافه کنید:

```
Sub Command1_Click ()
    If is_max((Form2.hWnd)) Then
       Label1.Caption = "Form 2 will be Maximized"
    Else
       Label1.Caption = "Form 2 will be Normalized"
    End If
End Sub
```

☜ توجه : Form2.hwnd را نمی توان مستقیماً بصورت یک پارامتر به یک تابع ارسال کرد. باید آنرا در یک سری پرانتز اضافه محصور کرد یا مقدارش را در یک متغیر موقتی ذخیره نمود.

۶- از منوی File گزینه Save Project را انتخاب نمایید تا فرمها و پروژه روی دیسک ذخیره شوند.

۷- برنامه را با فشار کلید F5 یا انتخاب گزینه Start از منوی Run اجرا کنید. Form2 را مینیمایز کنید، روی Command Button کلیک کنید و نتیجه ظاهر شده در Label1 را ملاحظه کنید. سپس Form2 را به حال اول بازگردانید، آنرا ماکزیمایز نموده و سپس

دوباره مینیمایز کنید. اکنون روی Command Button کلیک نمایید و نتیجه را بررسی کنید.

ایجاد پنجره Child مخفی

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
115781.TXT	۷ ژانویه ۱۹۹۹	3	3.1

برخلاف پنجره های معمولی، پنجره های MDI Child را نمی توان بدون ظاهر کردن آنها، به حافظه بار کرد. با این حال می توانید به کمک روش بیان شده در این بخش این کار را انجام دهید. در این روش از آرایه ای از فرمها برای مخفی کردن آنها استفاده می شود. برای سریعتر کردن نمایش فرمها یا انجام تغییرات در فرمها قبل از نمایش آنها، می توانید از این تکنیک استفاده نمایید.

مثال زیر نشان می دهد که چگونه:

- به کمک یک فرم MDI Child زمینه یک فرم MDI Parent را شبیه سازی نمایید و سپس دیگر فرمهای Child را در زیر آن فرم (ترتیب قرار گیری در عمق یا محور Z) قرار دهید تا مخفی شوند.
- از آرایه ای از فرمها برای کنترل وضعیت فرمها (که کدام یک از فرمها قبلا بار شده اند) استفاده نمایید.

این مثال یک فرم MDI Child (به نام BackSim) ایجاد می کند تا زمینه فرم MDI Parent را شبیه سازی نماید. هدف اصلی فرم BackSim عبارتست از مخفی کردن دیگر فرمهای Child که در زیر آن قرار می گیرند. به این ترتیب می توان فرمهای دیگر را در حافظه بار کرد بدون اینکه دیده شوند. مثال مزبور خصوصیات زیر را به فرم BackSim می دهد:

خصوصیت	مقدار
Enabled	False
ControlBox	False
Caption	""
BorderStyle	0 None
MaxButton	False
MinButton	False

در واقعه Resize از MDI Parent باید دستوراتی قرار دهید که باعث شوند فرم BackSim دقیقاً همان ابعاد و اندازه های فرم MDI Parent را داشته باشد. این کار باعث می شود که فرم Backsim مشابه فرم Parent به نظر برسد.

```
Sub MDIForm_Resize ()
   backsim.Move 0, 0, MDIform1.ScaleWidth, MDIform1.ScaleHeight
End Sub
```

به جای اینکه مطابق معمول مثال را بصورت یک سری مراحل قدم به قدم ارایه دهیم، روش همیشگی را طوری تغییر داده ایم تا بتوانید ساده تر این برنامه ویژوال بیسیک را ایجاد کنید. بنابراین، چهار فایل مورد نیاز یعنی MDI_SIM.BAS ، MDIFORM1.FRM ، BACK_SIM.FRM و SIMCHILD.FRM را در زیر آورده ایم تا شما بتوانید با کپی کردن متن مربوط به هر فایل در یک ویرایشگر متنی و ذخیره فایل ها، هر فایل را جداگانه ایجاد نمایید. روش استفاده از هر فایل در درون همان فایل بصورت توضیحات آورده شده است.

فرم MDI Parent

☜ توجه : این فرم در یک کامپیوتر با کیفیت تصویر ۷۶۸×۱۰۲۴ ایجاد شده است، بنابراین در صورت نیاز می توانید بعضی از خصوصیات فرم مانند ابعاد را برای کیفیت تفکیک پایینتر تنظیم کنید. این کار در داخل محیط برنامه نویسی ویژوال بیسیک قابل انجام است.

سطرهای زیر را در فایلی به نام MDIFORM1.FRM وارد کنید:

```
' The following includes the form and control descriptions as well as
' necessary Function and Sub procedures. Place this code in a single text
' file called MDIFORM1.FRM. so you can load it as a form in Visual Basic.
'
```

```
' NOTE: To make the code fit in this article, some of the lines
' are listed using multiple lines. After copying the code into a file,
' modify it to ensure that each line of code exists as one, single line
' in the file. Otherwise, you will receive errors when loading the form
' in Visual Basic.

VERSION 2.00
Begin MDIForm MDIForm1
   Caption         =   "MDIForm1"
   ClientHeight    =   5412
   ClientLeft      =   948
   ClientTop       =   1908
   ClientWidth     =   7368
   Height          =   6156
   Left            =   900
   LinkTopic       =   "MDIForm1"
   Tag             =   "Parent"
   Top             =   1212
   Width           =   7464
   Begin Menu mnu_children
      Caption         =   "&Children"
      Begin Menu mnu_newchild
         Caption         =   "&New Child"
         Shortcut        =   ^N
      End
      Begin Menu mnu_unloadall
         Caption         =   "&Unload All Children"
         Shortcut        =   ^U
      End
   End
   Begin Menu mnu_Window
      Caption         =   "&Window"
      Begin Menu mnu_cascade
         Caption         =   "&Cascade"
         Shortcut        =   ^C
      End
      Begin Menu mnu_tileh
         Caption         =   "Tile &Horizontal"
         Shortcut        =   ^H
      End
      Begin Menu mnu_tilev
         Caption         =   "Tile &Vertical"
         Shortcut        =   ^V
      End
      Begin Menu mnu_sim
         Caption         =   "Simulated &Background"
         Index           =   0
         Shortcut        =   ^B
      End
      Begin Menu mnu_showall
         Caption         =   "Show &All Children"
         Shortcut        =   ^A
      End
   End
End

Sub MDIForm_QueryUnload (Cancel As Integer, unloadmode As Integer)
   ' Unload all child forms loaded at runtime:
   mnu_unloadall_Click
```

```
End Sub

Sub MDIForm_Resize ()
   ' This reference to the properties of the BackSim
   ' MDI child form causes Visual Basic to implicitly load
   ' the form. Because MDI child forms cannot be loaded
   ' without being visible, this shows the BackSim form:
   backsim.Move 0, 0, MDIform1.ScaleWidth, MDIform1.ScaleHeight
End Sub

Sub mnu_cascade_Click ()
   backsim.ZOrder 1
   MDIform1.Arrange 0 ' cascade
   tiledflag = 0
End Sub

Sub mnu_newchild_Click ()
   ' Increment counter for array of child forms:
   childcount = childcount + 1

   ' Decide if new array slots are needed
   ' or if old (unloaded) slots need to be filled
   ' First, do empty slots exist:
   If childcount <= trackcount Then
      ' Find empty slot in arrays (menu and childtrack):
      lowestfreenum = 1
      For i = 1 To UBound(childtrack)
         If Not childtrack(i) Then
         lowestfreenum = i
         Exit For
         End If
      Next i
      ' Mark slot as loaded:
      childtrack(lowestfreenum) = True
      ' Load a new menu item for each child form
      ' under the menu control mnu_sim(0):
      Load mnu_sim(lowestfreenum)

      ' Set a new caption for the new menu item and supply access key:
      ' Turn the following two lines into one, single line:
      mnu_sim(lowestfreenum).Caption =
         "Show Only Child Form Number &" & lowestfreenum
      ' Mark this slot as loaded:
      childtrack(lowestfreenum) = True

      ' Set the tag equal to the childcount
      ' to allow tracking of self:
      child_array(lowestfreenum).Tag = lowestfreenum

      ' Set unique caption for each child form to cause an
      ' implicit load and show the child form:
      ' Turn the following two lines into one, single line:
      child_array(lowestfreenum).Caption =
         "Child Form Number " & lowestfreenum
   Else
      ' Create new slot:
      trackcount = trackcount + 1
      ' Increase size of both arrays:
      ReDim Preserve childtrack(1 To trackcount)
```

```
      ReDim Preserve child_array(1 To childcount)
      ' Load a new menu item for each child form
      ' under the menu control mnu_sim(0):
      Load mnu_sim(trackcount)

      ' Set a new caption for the new menu item and supply access key:
      ' Turn the following two lines into one, single line:
      mnu_sim(trackcount).Caption =
         "Show Only Child Form Number &" & trackcount

      ' Mark this slot as loaded:
      childtrack(trackcount) = True

      ' Set the tag equal to the childcount
      ' to allow tracking of self:
      child_array(trackcount).Tag = trackcount

      ' set unique caption for each child form
      ' this causes implicit load and shows child
      child_array(trackcount).Caption = "Child Form Number " & trackcount

   End If
End Sub

Sub mnu_showall_Click ()
   ' Place simulated background form at the bottom of the Z order:
   backsim.ZOrder 1

   ' Bring all the loaded child forms to the top of the Z order in
   ' succession:
   For i = 1 To trackcount
      If childtrack(i) Then
         If Not child_array(i).Enabled Then child_array(i).Enabled = True
         child_array(i).ZOrder 0
         child_array(i).Caption = child_array(i).Caption
      End If
   Next i
   Exclusive = False
End Sub

Sub mnu_sim_Click (index As Integer)

   ' Special case: if the arrange method has tiled the child windows,
   ' reset arrangement to cascade to get ZOrder method to work:
   If tiledflag Then
      MDIform1.Arrange 0  ' cascade
      tiledflag = False
   End If

   If index = 0 Then
      ' User wants to clear all children from MDI form
      ' but does not want to unload them:
      backsim.ZOrder 0
      ' Disable minimized form to avoid system menu popup:
      For i = 1 To trackcount 'UBound(childtrack)
         If childtrack(i) Then
            If child_array(i).WindowState = 1 Then
               child_array(i).Enabled = False
            End If
```

```
         End If
      Next i
   Else

      ' Bring the simulated background to the top
      ' of the Z order to hide all other children:
      backsim.ZOrder 0
      ' Disable minimized form to avoid system menu popup:
      For i = 1 To UBound(childtrack)
         If childtrack(i) Then
            If child_array(i).WindowState = 1 Then
               child_array(i).Enabled = False
            End If
         End If
      Next i

      ' Bring desired child form to the top:
      child_array(index).ZOrder 0
      child_array(index).Enabled = True
      child_array(index).Caption = child_array(index).Caption
      child_array(index).SetFocus
      ' Set flag for QueryUnload event of child form
      ' to avoid a repaint over the next child form in
      ' Z order if unload current "only" child:
      Exclusive = True
   End If
End Sub

Sub mnu_tileh_Click ()
   backsim.ZOrder 1
   MDIform1.Arrange 1 ' Tile horizontal.
   tiledflag = 1
End Sub

Sub mnu_tilev_Click ()
   backsim.ZOrder 1
   MDIform1.Arrange 2 ' Tile vertical.
   tiledflag = 2
End Sub

Sub mnu_unloadall_Click ()
   ' Unload all child forms marked
   ' as loaded in childtrack array:
   For i = 1 To trackcount ' UBound(childtrack)
      If childtrack(i) Then
         Unload child_array(i)
      End If
   Next i
   ' Reset counters:
   trackcount = 0
   childcount = 0
End Sub
```

سطرهای زیر را در فایل BACK_SIM.FRM وارد کنید:

```
' The following includes the form and control descriptions as well
' as necessary Function and Sub procedures. Place the code in a
' single text file called BACK_SIM.FRM, so you can load it as a form
' in Visual Basic.

VERSION 2.00
Begin Form backsim
   BackColor       =   &H00C0C0C0&
   BorderStyle     =   0  'None
   ClientHeight    =   4776
   ClientLeft      =   1632
   ClientTop       =   1644
   ClientWidth     =   7368
   ControlBox      =   0   'False
   Enabled         =   0   'False
   Height          =   5196
   Left            =   1584
   LinkTopic       =   "Form2"
   MaxButton       =   0   'False
   MDIChild        =   -1  'True
   MinButton       =   0   'False
   ScaleHeight     =   4776
   ScaleWidth      =   7368
   Tag             =   "BackSim"
   Top             =   1272
   Width           =   7464
End
Sub Form_Load ()
   ' Set the backcolor property to match
   ' whatever system color setting the user has set
   ' for the Background color of multiple document
   ' interface (MDI) applications:
   Me.BackColor = GetSysColor(COLOR_APPWORKSPACE)
End Sub
```

سطرهای زیر را در فایل SIMCHILD.FRM وارد کنید:

```
' The following includes the form and control description as well as
' necessary Function and Sub procedures. Place the code in a single text
' file called SIMCHILD.FRM, so you can load it as a form in Visual Basic.

ERSION 2.00
Begin Form ftemplate
   Caption         =   "ftemplate"
   ClientHeight    =   6300
   ClientLeft      =   1116
   ClientTop       =   1224
   ClientWidth     =   7368
   Height          =   6720
   Left            =   1068
   LinkTopic       =   "Form3"
   MDIChild        =   -1  'True
   ScaleHeight     =   6300
   ScaleWidth      =   7368
   Top             =   852
   Width           =   7464
End
```

```
Sub Form_QueryUnload (cancel As Integer, unloadmode As Integer)
   ' Remove menu item pointing to Me:
   Unload mdiform1.mnu_sim(Val(Me.Tag))
   ' Mark my slot as unloaded:
   childtrack(Val(Me.Tag)) = False
   ' Decrement total childcount:
   childcount = childcount - 1

   ' If this form was the only child form visible
   ' at the time of unload, put background
   ' form the top of the z order:
   If Exclusive Then
      ' Must enable background form temporarily
      ' so that it will be the next in the Z order
      ' after the unload. This avoids a repaint over
      ' the previous child in the Z order.
      backsim.Enabled = True
      backsim.ZOrder 0
      backsim.Enabled = False
      Exclusive = False
   End If
End Sub

Sub Form_Resize ()
   ' Keep minimized children visible by
   ' setting the ZOrder explicitly:
   If Me.WindowState = 1 Then
      Me.ZOrder 0
      ' Force repaint of caption:
      Me.Caption = Me.Caption
   End If
End Sub
```

روش ایجاد برنامه و اجرای آن:

۱- در ویژوال بیسیک یک پروژه جدید ایجاد کنید. Form1 بطور پیش فرض ایجاد می شود.

۲- از منوی File گزینه Remove File را انتخاب کنید و Form1 را به کمک آن از پروژه حذف کنید.

۳- از منوی File گزینه Add Form را انتخاب کرده و فرم MDIFORM1.FRM را به پروژه اضافه نمایید.

۴- مرحله ۳ را برای فرمهای BACKSIM.FRM و SIMCHILD.FRM تکرار نمایید.

۵- از منوی Options گزینه Project را انتخاب نموده و فرم راه انداز[1] برنامه را MDIFORM1 تعیین نمایید.

---

[1] Start up Form

۶- یک ماژول جدید در پروژه تعریف کنید (MDI_SIM.BAS) و کدهای زیر را به بخش declarations این ماژول اضافه نمایید:

```
Global child_array() As New ftemplate
Global childtrack() As Integer
Global childcount As Integer, trackcount As Integer
Global lowestfreenum As Integer, i  As Integer
Global Exclusive As Integer, tiledflag As Integer

Declare Function GetSysColor Lib "User" (ByVal nIndex%) As Long

Global Const COLOR_APPWORKSPACE = 12
```

پروژه را ذخیره کرده و برنامه را اجرا نمایید. تمامی گزینه های منو را آزمایش کنید، فرمهای Child را ببندید یا مینیمایز کنید. توجه کنید که برنامه، وضعیت نمایش فرمها را تحت کنترل دارد و از مکانهایی که در آرایه فرمها خالی می شود دوباره استفاده می کند. فرمهای Child در این مثال برمبنای فرم SIMCHILD.FRM که متغیر ftemplate به آن اشاره می کند، ایجاد می شوند. می توانید این متغیر را طوری تغییر دهید که از فرم مورد نظر شما استفاده شود. در هر حال بررسی کنید آیا توانایی مخفی کردن پنجره های Child ارزش نوشتن کدهای اضافه را دارد یا خیر!

جلوگیری از جابجایی و تغییر اندازه پنجره ها

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
118376.TXT	۲۱ سپتامبر ۱۹۹۸	2 , 3	3.1

در این بخش خواهیم دید که چگونه گزینه های تغییر اندازه را از منوی کنترل (منوی سیستم) پنجره های ویژوال بیسیک حذف کنیم تا از امکان تغییر محل یا تغییر اندازه پنجره ها

جلوگیری نماییم.

اگر مایل نیستید کاربر برنامه شما قادر باشد پنجره برنامه را تغییر اندازه داده یا جابجا کند، با حذف گزینه های Move و یا Size از منوی کنترل پنجره که توسط فراخوانی توابع API صورت می گیرد می توانید به این هدف برسید. تابع GetSystemMenu می تواند شناسه مربوط به منوی کنترل را مشخص کند. سپس می توانید با استفاده از این شناسه و به کمک تابع DeleteMenu گزینه های خاصی را در این منو تغییر داده یا حذف کنید.

مراحل زیر نشان می دهند چگونه گزینه های Move و Size را از منوی کنترل حذف کنید:

۱- یک پروژه جدید در ویژوال بیسیک ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- یک Command Button روی Form1 قرار دهید. عنوان این دکمه را به Lock Form تغییر دهید.

۳- لیست زیر را به بخش declarations از Form1 اضافه نمایید:

```
' Enter each Declare statement as one, single line:
Declare Function GetSystemMenu Lib "User" (ByVal hWnd%,
      ByVal bRevert%) As Integer
Declare Function DeleteMenu Lib "user" (ByVal hMenu%, ByVal iditem%,
      ByVal wflags%) As Integer

Const SC_SIZE = &HF000
Const SC_MOVE = &HF010
Const MF_BYCOMMAND = &H0
```

☜ توجه : ثابتهایی که برای حذف گزینه های دیگر از منوی سیستم بکار می روند در فایل WIN30API.TXT که در شاخه \VB\WINAPI\ قرار دارد معرفی شده اند.

۴- کد زیر را به واقعه Click از Command1 اضافه نمایید:

```
Sub Command1_Click ()
    Dim hWnd%, hMenu%, Success%
    hWnd% = Form1.hWnd
    hMenu% = GetSystemMenu(hWnd%, 0)

    Success% = deletemenu(hMenu%, SC_SIZE, MF_BYCOMMAND)
    Success% = deletemenu(hMenu%, SC_MOVE, MF_BYCOMMAND)
End Sub
```

۵- برای اجرای برنامه F5 را فشار دهید.

روی Command Button کلیک کنید تا گزینه های منو حذف شوند. اکنون سعی کنید تا

پنجره را حرکت داده یا تغییر اندازه دهید. ملاحظه می کنید که قادر به این کارها نخواهید بود. با این حال، می توانید پنجره را مینیمایز یا ماکزیمایز کرده یا از طریق کد، سایز و مکان پنجره را تغییر دهید.

روش دیگری که مشابه این روش است در بخشی از بانک اطلاعاتی مایکروسافت تحت شماره ۸۲۸۷۶ ذخیره شده است.

نمایش علامت تغییر اندازه پنجره ها در ویندوز ۹۵ / ۹۸

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
143262.TXT	۲۴ آگوست ۱۹۹۸	4 , 5 , 6	95 / 98

دو راه برای فعال کردن علامت تغییر اندازه پنجره ها در ویندوز ۹۵/۹۸ که در گوشه سمت راست پایین پنجره ظاهر می شود، وجود دارد. این علامت که در بعضی پنجره ها ظاهر می شود نشانه اینست که پنجره مذکور قابلیت تغییر اندازه را داراست.

روش اول استفاده از ابزار Status Bar است که به همراه ابزارهای عمومی ویندوز (MSCOMCTL.OCX) ارایه می شود. این ابزار در پایین پنجره قرار می گیرد و اگر پنجره قابل تغییر اندازه باشد، علامت تغییر اندازه پنجره به طور خودکار روی این ابزار در سمت راست ظاهر می شود.

روش دوم عبارتست از قرار دادن یک TextBox روی پنجره به طوری که گوشه سمت راست پایین TextBox روی گوشه سمت راست پایین پنجره منطبق باشد. در ضمن لازمست خصوصیات TextBox به این ترتیب تنظیم شوند:

```
Text1.Multiline = True
Text1.Scrollbars = 3    'Both
```

کد زیر نحوه انجام این کار را نشان می دهد:

```
Const x = 100       'where x and y are scaled values that you
Const y = 385       'use to take into account the fact that
                    'the TextBox should be a little less wide
                    'and high than the form. You should
                    'initialize these to constant values.

Private Sub Form_Load()
    Text1.Left = 0
    Text1.Top = 0
    Text1.Width = Me.Width - x
    Text1.Height = Me.Height - y
End Sub

Private Sub Form_Resize()
    Text1.Width = Me.Width - x
    Text1.Heigh
   ht = Me.Height - y
End Sub
```

ایجاد نوار عنوان چشمک زن روی پنجره

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
147815.TXT	۷ آگوست ۱۹۹۸	4 , 5 , 6	95 / 98

به کمک تابع FlashWindow می توانید نوار عنوان هر پنجره ای را که شناسه آنرا بدانید به حالت چشمک زن در آورید. ویژوال بیسیک این توانایی را دارد که نوار عنوان هر پنجره ای را که شناسه آن را در اختیار داشته باشد به حالت چشمک زن در آورد. تابع FlashWindow عمل چشمک زدن پنجره مورد نظر را فقط یکبار انجام می دهد. چشمک زدن یک پنجره به معنی تغییر دادن نحوه نمایش نوار عنوان آن پنجره است، به این صورت که پنجره از حالت فعال به حالت غیر فعال در بیاید (یا بر عکس) و سپس به حال اول خود برگردد.

اصولاً از چشمک زدن پنجره به این منظور استفاده می کنیم که به کاربر اطلاع دهیم باید به این پنجره توجه کند و این زمانی است که تمرکز روی این پنجره نباشد. تابع FlashWindow به

صورت زیر تعریف می شود:

```
FlashWindow(hWnd, bInvert)
```

که در آن :

hWnd = مشخص کننده پنجره ای است که باید چشمک بزند. این پنجره می تواند باز بوده یا بصورت یک آیکون باشد.

bInvert = مشخص می کند که پنجره باید به حالت چشمک زده در آید یا به وضعیت اولیه خود بر گردد. اگر پارامتر bInvert صفر نباشد پنجره به حالت چشمک زده در می آید. اگر bInvert صفر باشد پنجره به حالت اولیه خود (فعال یا غیر فعال)در می آید.

تابع FlashWindow عددی را بر می گرداند که نشان دهنده وضعیت پنجره قبل از فراخوانی این تابع بوده است. اگر پنجره فعال بوده، مقدار برگشتی غیر صفر و در غیر اینصورت صفر خواهد بود.

هنگام استفاده از FlashWindow خوبست که سرعت چشمک زدن را معادل سرعت چشمک زدن مکان نما تنظیم کنید. سرعت چشمک زدن مکان نما را می توان از طریق تابع GetCaretBlinkTime پیدا کرد. این تابع زمان لازم برای چشمک زدن مکان نما را بر حسب میلی ثانیه (هزارم ثانیه) مشخص می کند.

مثال زیر نشان می دهد چگونه می توانید پنجره ای را که کنترل اجرای برنامه روی آن متمرکز نیست، به حالت چشمک زن در بیاورید:

۱- یک پروژه جدید در ویژوال بیسیک ایجاد کنید. Form1 بصورت پیش فرض ایجاد می شود.

۲- یک Timer به Form1 اضافه کنید.

۳- کد زیر را در Form1 وارد کنید:

```
Private Sub Form_GotFocus()
   Timer1.Enabled = False
End Sub

Private Sub Form_Load()
   Timer1.Interval = GetCaretBlinkTime()
   Form2.Show
End Sub

Private Sub Timer1_Timer()
   Success = FlashWindow(Form1.hwnd, 1)
End Sub
```

۴- فرم دیگری به پروژه اضافه کنید.

۵- کد زیر را در Form2 وارد کنید:

```
Private Sub Form_Click()
   Form1.Timer1.Enabled = True
End Sub
```

۶- یک ماژول به پروژه اضافه کنید.

۷- کد زیر را در Module1 وارد نمایید:

```
'Depending on what operating system you are using determines the
'correct function declares and variables. This is an example of
'conditional compilation.

#If Win32 Then
   Declare Function FlashWindow Lib "user32" (ByVal hwnd As Long,_
                       ByVal bInvert As Long) As Long
   Declare Function GetCaretBlinkTime Lib "user32" () As Long
   Dim Success As Long
#Else
   Declare Function FlashWindow Lib "User" (ByVal hwnd As Integer, _
                      ByVal bInvert As Integer) As Integer
   Declare Function GetCaretBlinkTime Lib "User" () As Integer
   Dim Success As Integer
#End If
```

۸- در منوی Run گزینه Start را انتخاب کنید یا کلید F5 را فشار دهید تا برنامه اجرا شود. Form1 در جلو ظاهر می شود و Form2 در پشت آن. روی Form2 کلیک کنید، نوار عنوان Form1 شروع به چشمک زدن می کند و آنقدر ادامه می دهد تا اینکه روی Form1 کلیک کنید.

جابجایی پنجره ای که نوار عنوان ندارد

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
173773.TXT	۷ آگوست ۱۹۹۸	4 , 5 , 6	95 / 98

در این بخش خواهیم دید که چگونه می توان پنجره ای را که نوار عنوان ندارد، توسط ماوس

جابجا نمود. این کار به کمک تابع SendMessage و پیام WM_NCLBUTTONDOWN انجام می شود. هنگامی که کاربر روی نوار عنوان یک پنجره با دکمه سمت راست ماوس کلیک می کند پیام WM_NCLBUTTONDOWN توسط ویندوز ایجاد می شود. این پیام برای تعیین اینکه آیا پنجره مورد نظر در حال جابجا شدن می باشد یا خیر بکار می رود. اگر نوار عنوان وجود نداشته باشد، این پیام تولید نخواهد شد. اغلب نیاز است که پنجره ای که نوار عنوان ندارد را جابجا کنیم. پنجره یا فرم در ویژوال بیسیک ۳۲ بیتی در موارد زیر نوار عنوان نخواهد داشت :

در ویژوال بیسیک ۵ و ٦

```
Caption = ""
ControlBox = False
```

در ویژوال بیسیک ۴

```
Caption = ""
ControlBox = False
MaxButton = False
MinButton = False
```

مثال زیر روشی را نشان می دهد که بوسیله آن، کاربر می تواند پنجره ای بدون نوار عنوان را با کلیک کردن روی هر قسمتی از پنجره و حرکت دادن ماوس، جابجا نماید. در صورت نیاز می توان کدی که در واقعه MouseMove از Form1 قرار دارد را در دیگر واقعه ها قرار داد. به عنوان مثال، کد مزبور را می توان در واقعه MouseDown مربوط به یک Label قرار داد تا عمل جابجایی پنجره توسط کلیک کردن روی آن Label انجام شود.

☜ توجه: در ویژوال بیسیک ۴ پنجره ای که دارای منو باشد، نوار عنوان نیز خواهد داشت بنابراین به روش گفته شده در زیر نیازی نخواهد داشت. در ویژوال بیسیک ۵ و ٦، پنجره ای که دارای منو است می تواند نوار عنوان نداشته باشد.

۱- یک پروژه جدید ایجاد کنید. Form1 بطور پیش فرض بوجود می آید.

۲- خصوصیات زیر را برای Form1 تنظیم کنید:

خصوصیت	مقدار
Caption	""
ControlBox	False
MinButton	False
MaxButton	False

۳- یک CommandButton روی Form1 قرار دهید.

۴- کد زیر را در بخش Declarations از Form1 وارد کنید:

```
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 Declare Sub ReleaseCapture Lib "User32" ()

Const WM_NCLBUTTONDOWN = &HA1
Const HTCAPTION = 2

Private Sub Form_Load()
    Command1.Caption = "Exit"
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
                           X As Single, Y As Single)
    Dim lngReturnValue As Long
    If Button = 1 Then
        Call ReleaseCapture
        lngReturnValue = SendMessage(Form1.hWnd, WM_NCLBUTTONDOWN, _
                                                        HTCAPTION, 0&)
    End If
End Sub

Private Sub Command1_Click()
    End
End Sub
```

۵- برای اجرای برنامه کلید F5 را فشار دهید. برای جابجایی پنجره می توانید روی هر قسمت از آن کلیک کرده و ماوس را حرکت دهید. با کلیک کردن روی CommandButton از برنامه خارج می شوید.

محدود کردن میزان جابجایی ماوس در داخل یک پنجره

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
179192.TXT	۷ آگوست ۱۹۹۸	5 , 6	95 / 98

تابع ClipCursor این امکان را بوجود می آورد تا بتوانید جابجایی نشانگر ماوس را در یک بخش خاص، مثلاً یک پنجره، محدود کنید. این بخش روش استفاده از این تابع را طی یک مثال نشان می دهد. نمونه ای از کاربرد این روش زمانی است که در برنامه ای از قابلیت Drag and Drop استفاده می کنید و مایلید امکان رها کردن آیتم ها فقط در داخل یک فرم بخصوص وجود داشته باشد.

تابع ClipCursor مکان نمای ماوس را در یک محدوده مستطیل شکل روی صفحه محدود می‌کند. سپس اگر نشانگر ماوس بخواهد از این محدوده خارج شود (چه توسط حرکت دادن ماوس و چه توسط تابع SetCursorPos)، ویندوز به طور خودکار موقعیت مکان نما را طوری تنظیم می کند تا در درون مستطیل مزبور قرار گیرد.

لبه های پنجره به عنوان محدوده مستطیل شکل برای تابع ClipCursor بکار می روند. برای تعیین اندازه پنجره از تابع GetClientRect استفاده نمایید. این تابع مختصات محدوده بیرونی پنجره را مشخص می کند. این مختصات نسبت به گوشه سمت چپ بالای محدوده بیرونی پنجره تعیین می شوند که نقطه (0,0) است. این بدین معناست که شما مجبورید این مختصات را به مختصات صفحه تبدیل نمایید.

تابع ClientToScreen مختصات یک نقطه از محدوده مذکور را به مختصات صفحه تبدیل می نماید. آخرین تابع مورد استفاده عبارتست از OffsetRect . تابع OffsetRect مستطیل مشخصی را به میزان دلخواه جابجا می نماید.

☜ توجه: اگر پس از فراخوانی تابع ClipCursor پنجره را Unload کنید، مکان نما همچنان در محدوده آخرین مکان پنجره محصور خواهد بود. برای جلوگیری از این اتفاق، لازمست در واقعه Form_Unload تابع ClipCursor را توسط پارامتر تهی (Null) فرابخوانید.

مراحل ایجاد مثال به صورت زیر می باشد:

۱- یک پروژه جدید در ویژوال بیسیک ایجاد نمایید. Form1 بطور پیش فرض ایجاد می شود.

۲- یک CommandButton به Form1 اضافه کنید.

۳- کد زیر را در Form1 وارد کنید:

```
Option Explicit

Private Type RECT
    left As Integer
    top As Integer
    right As Integer
    bottom As Integer
End Type

Private Type POINT
    x As Long
    y As Long
End Type

Private Declare Sub ClipCursor Lib "user32" (lpRect As Any)
Private Declare Sub GetClientRect Lib "user32" _
                                          (ByVal hWnd As Long, _
                                           lpRect As RECT)
Private Declare Sub ClientToScreen Lib "user32" _
                                          (ByVal hWnd As Long, _
                                           lpPoint As POINT)
Private Declare Sub OffsetRect Lib "user32" (lpRect As RECT, _
                                        ByVal x As Long, _
                                        ByVal y As Long)
Private Sub Form_Load()
    Command1.Caption = "Limit Cursor Movement"
    Command2.Caption = "Release Limit"
End Sub

Private Sub Command1_Click()
    'Limits the Cursor movement to within the form.
    Dim client As RECT
    Dim upperleft As POINT

    GetClientRect Me.hWnd, client
    upperleft.x = client.left
    upperleft.y = client.top
    ClientToScreen Me.hWnd, upperleft
    OffsetRect client, upperleft.x, upperleft.y
    ClipCursor client
End Sub

Private Sub Command2_Click()
    'Releases the cursor limits
    ClipCursor ByVal 0&
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'Releases the cursor limits
```

```
    ClipCursor ByVal 0&
End Sub
```

۴- از منوی Run گزینه Start را انتخاب کنید یا کلید F5 را فشار دهید تا برنامه اجرا شود. روی Limit Cursor Movement کلیک کنید و توجه کنید که حرکت مکان نمای ماوس در لبه های پنجره محدود می شود. روی Release Limit کلیک کنید تا مکان نما بتواند در تمامی صفحه حرکت کند.

ایجاد پنجره ای که همیشه در جلوی بقیه قرار می گیرد

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
184297.TXT	۱۳ آگوست ۱۹۹۸	4 , 5 , 6	95 / 98

ویژوال بیسیک روش یا خصوصیت ویژه ای ندارد که باعث شود یک پنجره همیشه جلوی دیگر پنجره ها قرار بگیرد. این کار توسط تابع SetWindowPos صورت می گیرد. در این بخش خواهیم دید که چگونه توسط SetWindowPos پنجره ای را طوری تنظیم کنیم که همیشه در جلوی بقیه قرار بگیرد.

مثال زیر از تابعی به نام SetTopMostWindow استفاده می کند. این تابع می تواند پنجره ای را بصورت "همیشه در جلو" یا بصورت معمولی در آورد. این کار به کمک دو پارامتر hwnd و Topmost صورت می گیرد.

پارامتر hwnd مشخص کننده شناسه پنجره ای است که قرار است بصورت "همیشه در جلو" یا بصورت معمولی در بیاید.

پارامتر TopMost مشخص می کند که پنجره بصورت "همیشه در جلو" یا بصورت معمولی درآید. اگر مقدار این پارامتر True باشد، تابع مذکور باعث می شود که پنجره مورد نظر همیشه در جلوی دیگر پنجره ها باشد. اگر مقدار این پارامتر False باشد، پنجره مورد نظر بصورت

معمولی در خواهد آمد.

۱- یک پروژه جدید ایجاد کنید. Form1 بصورت پیش فرض ایجاد خواهد شد.

۲- دو CommandButton به Form1 اضافه کنید(Command1 و Command2).

۳- عنوان Command1 را معادل Always on top تنظیم کنید.

۴- عنوان Command2 را معادل Normal تنظیم کنید.

۵- کد زیر را در بخش Declarations از Form1 وارد کنید:

```
Option Explicit

Private Sub Command1_Click()
    Dim lR As Long
    lR = SetTopMostWindow(Form1.hwnd, True)
End Sub

Private Sub Command2_Click()
    Dim lR As Long
    lR = SetTopMostWindow(Form1.hwnd, False)
End Sub
```

٦- در منوی Project گزینه Add Module را انتخاب کنید تا یک ماژول جدید به پروژه اضافه شود.

۷- کد زیر را به ماژول جدید اضافه کنید:

```
Option Explicit
Public Const SWP_NOMOVE = 2
Public Const SWP_NOSIZE = 1
Public Const FLAGS = SWP_NOMOVE Or SWP_NOSIZE
Public Const HWND_TOPMOST = -1
Public Const HWND_NOTOPMOST = -2

Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos"  _
           (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

Public Function SetTopMostWindow(hwnd As Long, Topmost As Boolean) _
         As Long

    If Topmost = True Then 'Make the window topmost
        SetTopMostWindow = SetWindowPos(hwnd, HWND_TOPMOST, 0, 0, 0, _
               0, FLAGS)
    Else
        SetTopMostWindow = SetWindowPos(hwnd, HWND_NOTOPMOST, 0, 0, _
               0, 0,FLAGS)
```

```
        SetTopMostWindow = False
    End If
End Function
```

توجه : در مثال فوق، علامت "زیرخط" (_) نشان دهنده ادامه سطر می باشد.

۸- برای اجرای برنامه کلید F5 را فشار دهید.

اگر روی Always on top کلیک کنید، پنجره شما بصورت "همیشه در جلو" در خواهد آمد و روی بقیه پنجره ها قرار می گیرد، به این ترتیب این امکان وجود ندارد که هیچ پنجره دیگری را روی این پنجره قرار دهید. اگر روی Normal کلیک کنید، فرم به حالت معمولی بر می گردد (یعنی می توانید پنجره های دیگری روی آن قرار دهید).

محدود کردن اندازه حداقل و حداکثر یک پنجره

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
185733.TXT	۱۷ آگوست ۱۹۹۸	5 , 6	95 / 98

ویژوال بیسیک این امکان را فراهم می آورد که یک پنجره را به هر اندازه ای که بخواهید کوچک و بزرگ کنید (در صورتیکه خصوصیت BorderStyle را بصورت "قابل تغییر اندازه" تعریف کرده باشید). در مواردی ممکنست بخواهید (یا لازم باشد) محدوده ای برای تغییر اندازه پنجره تعریف کنید (مثلا از حدی کوچکتر یا بزرگتر نشود).

به عنوان مثال، برنامه Microsoft Paint اجازه نمی دهد پنجره اصلی این برنامه از حدی کوچکتر شود. به این ترتیب کاربر قادر خواهد بود همیشه حداقل بخشی از تصویری که درحال ویرایش آن است را مشاهده کند.

با کمک توابع API می توانید عملکردی مشابه برنامه مذکور را در برنامه های ویژوال بیسیک خود بوجود آورید.

اخطار : هر گونه استفاده از مثال زیر با مسئولیت خودشما خواهد بود.

مراحل ایجاد مثال مورد بحث را در زیر ملاحظه می نمایید. برای اینکه به نتیجه مطلوب برسیم لازمست از مفهومی به نام "کنترل پیام" استفاده گردد تا وقوع پیام WM_GETMINMAXINFO قابل تشخیص باشد. این پیام زمانی ارسال می شود که کسی سعی کند پنجره ای را تغییر اندازه دهد.

برای کسب اطلاعات بیشتر در این زمینه میتوانید به مورد شماره 168795 تحت عنوان "کنترل پیامهای ارسالی ویندوز به کمک AddressOf" نیز مراجعه نمایید.

لازمست پیش از پاک کردن یک پنجره از حافظه، عمل کنترل پیامهای مربوط به آن پنجره را خاتمه دهید. در صورتی که این کار انجام نگیرد ممکنست باعث بروز مشکل در برنامه شده، ایجاد خطای Invalid Page Fault نموده و درنهایت باعث از دست رفتن داده ها گردد. همیشه باید پیش از پاک کردن پنجره از حافظه یا خروج از برنامه، عملیات کنترل پیامهای پنجره را خاتمه دهید. این نکته خصوصاً زمان رفع اشکال از برنامه در داخل محیط ویژوال بیسیک بسیار مهم است. کلیک کردن روی دکمه توقف برنامه یا انتخاب گزینه End از منوی Run بدون خاتمه دادن به عملیات کنترل پیامها، باعث خطای Invalid Page Fault شده و ویژوال بیسیک را خواهد بست.

مراحل ایجاد مثال به این ترتیب است:

۱- یک پروژه جدید ایجاد کنید.

۲- کد زیر را در Form1 وارد نمایید:

```
Option Explicit

Private Sub Form_Load()
    'Save handle to the form.
    gHW = Me.hwnd

    'Begin subclassing.
    Hook
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'Stop subclassing.
    Unhook
End Sub
```

۳- یک ماژول جدید به پروژه اضافه نمایید.

۴- کد زیر را به ماژول جدید اضافه کنید:

```
  Option Explicit

  Private Const GWL_WNDPROC = -4
  Private Const WM_GETMINMAXINFO = &H24

  Private Type POINTAPI
      x As Long
      y As Long
  End Type

  Private Type MINMAXINFO
      ptReserved As POINTAPI
      ptMaxSize As POINTAPI
      ptMaxPosition As POINTAPI
      ptMinTrackSize As POINTAPI
      ptMaxTrackSize As POINTAPI
  End Type

  Global lpPrevWndProc As Long
  Global gHW As Long

Private Declare Function DefWindowProc Lib "user32" Alias _
   "DefWindowProcA" (ByVal hwnd As Long, ByVal wMsg As Long, _
    ByVal wParam As Long, ByVal lParam As Long) As Long
Private 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
Private Declare Function SetWindowLong Lib "user32" Alias _
   "SetWindowLongA" (ByVal hwnd As Long, _
    ByVal nIndex As Long, ByVal dwNewLong As Long) As Long
Private Declare Sub CopyMemoryToMinMaxInfo Lib "KERNEL32" Alias _
   "RtlMoveMemory" (hpvDest As MINMAXINFO, ByVal hpvSource As Long, _
    ByVal cbCopy As Long)
Private Declare Sub CopyMemoryFromMinMaxInfo Lib "KERNEL32" Alias _
   "RtlMoveMemory" (ByVal hpvDest As Long, hpvSource As MINMAXINFO, _
    ByVal cbCopy As Long)

  Public Sub Hook()
      'Start subclassing.
      lpPrevWndProc = SetWindowLong(gHW, GWL_WNDPROC, _
         AddressOf WindowProc)
  End Sub

  Public Sub Unhook()
      Dim temp As Long

      'Cease subclassing.
      temp = SetWindowLong(gHW, GWL_WNDPROC, lpPrevWndProc)
  End Sub

  Function WindowProc(ByVal hw As Long, ByVal uMsg As Long, _
     ByVal wParam As Long, ByVal lParam As Long) As Long
      Dim MinMax As MINMAXINFO

      'Check for request for min/max window sizes.
      If uMsg = WM_GETMINMAXINFO Then
          'Retrieve default MinMax settings
          CopyMemoryToMinMaxInfo MinMax, lParam, Len(MinMax)
```

```
            'Specify new minimum size for window.
            MinMax.ptMinTrackSize.x = 200
            MinMax.ptMinTrackSize.y = 200

            'Specify new maximum size for window.
            MinMax.ptMaxTrackSize.x = 500
            MinMax.ptMaxTrackSize.y = 500

            'Copy local structure back.
            CopyMemoryFromMinMaxInfo lParam, MinMax, Len(MinMax)

            WindowProc = DefWindowProc(hw, uMsg, wParam, lParam)
        Else
            WindowProc = CallWindowProc(lpPrevWndProc, hw, uMsg, _
               wParam, lParam)
        End If
    End Function
```

۵- مثال را ذخیره کرده و اجرا نمایید.

۶- سعی کنید پنجره را تغییر اندازه دهید. ملاحظه خواهید نمود که پنجره مجاز نیست به اندازه ای کوچکتر از ۲۰۰ در ۲۰۰ پیکسل یا بزرگتر از ۵۰۰ در ۵۰۰ پیکسل درآید.

قرار دادن پنجره یک برنامه در حال اجرا در جلوی بقیه

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
186431.TXT	۱۷ آگوست ۱۹۹۸	4 , 5 , 6	95 / 98

در این بخش خواهیم دید که چگونه از توابع BringToTop و FindWindow برای قرار دادن پنجره یک برنامه دیگر در جلوی بقیه و انتقال کنترل به آن استفاده نماییم.

۱- یک پروژه جدید در ویژوال بیسیک ایجاد نمایید. Form1 بطور پیش فرض ایجاد می شود.

۲- سه CommandButton روی این فرم قرار دهید. ابعاد Command2 و Command3 را بزرگتر اختیار کنید تا عنوان آنها قابل خوانده شدن باشد.

۳- کد زیر را در بخش Declarations از Form1 وارد کنید:

```
Option Explicit
Dim THandle As Long

Private Declare Function BringWindowToTop Lib "user32" (ByVal _
   hwnd As Long) As Long

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

Private Sub Form_Load()
   Command1.Caption = "Run the Calculator"
   Command2.Caption = "Find Calculator's Handle"
   Command3.Caption = "Bring Calculator To The Top"
   Command2.Enabled = False
   Command3.Enabled = False
End Sub

Private Sub Command1_Click()
   Dim X As Long
   X = Shell("Calc.exe", 1)
   If X <> 0 Then
      Command2.Enabled = True
   End If
End Sub

Private Sub Command2_click()
   THandle = FindWindow(vbEmpty, "Calculator")
   If THandle = 0 Then
      Command3.Enabled = False
      MsgBox "Sorry, the calculator is running"
      Exit Sub
   End If
   Command3.Enabled = True
End Sub

Private Sub Command3_click()
   Dim iret As Long
   iret = BringWindowToTop(THandle)
End Sub
```

۴- بـرنامه را اجرا کنید. هنگامی که روی Command1 کلیک می کنید، ماشین حساب ظاهر شـده و Command2 فعـال مـی شود. اندازه این برنامه (مثال) را ماکزیمایز کنید تا تمام صـفحه را بپوشـاند. اگر برنامه بتواند شناسه برنامه ماشین حساب را بیابد، Command3 فعال می شود.

۵- روی Command3 کـلیک نمایید، در این هنگام ماشین حساب به جلو آمده و کنترل را در اختیار خواهد گرفت.

پوشاندن روی Taskbar توسط یک پنجره

نام فایل	تاریخ	نگارش ویژوال بیسیک	نگارش ویندوز
197585.TXT	۱۷ دسامبر ۱۹۹۸	4 , 5 , 6	95 / 98

Taskbar ویندوز دارای خصوصیت AutoHide است که باعث می شود در لبه صفحه مخفی شده و فضای بسیار کمی اشغال نماید و این وضعیت را حفظ نماید تا اینکه ماوس روی آن قرار گرفته و باعث شود دوباره به وضعیت عادی خود برگردد.

زمانی که خصوصیت AutoHide را فعال می کنید، پنجره های ماکزیمایز شده تمام صفحه را می پوشانند. اما اگر AutoHide فعال نباشد، پنجره ماکزیمایز شده تمام صفحه به غیر بخشی که در اشغال Taskbar است را خواهد پوشانید.

هیچ راهی وجود ندارد که از طریق برنامه نویسی بتوان خصوصیت AutoHide را تغییر داد. برای اینکه بتوانید یک پنجره ویژوال بیسیک را طوری بزرگ کنید که تمام صفحه از جمله روی Taskbar را نیز بپوشاند (در صورتیکه Taskbar روی صفحه نمایش داده شده باشد)، باید طول و عرض پنجره را تنظیم کنید.

در مثال زیر خواهیم دید که چگونه تابع SetWindowPos پنجره مشخصی را قادر می سازد تمام صفحه از جمله تمام Taskbar ها را بپوشاند. این روش یا تمامی روشهای مشابه فقط زمانی عمل می کنند که پنجره قبلاً ماکزیمایز نشده باشد. اگر پنجره مورد نظر در حال حاضر ماکزیمایز شده باشد باید قبل از استفاده از تابع SetWindowPos ابتدا آنرا به حالت معمولی برگردانید.

۱- یک پروژه جدید ایجاد کنید.Form1 بطور پیش فرض بوجود می آید.

۲- یک CommandButton به Form1 اضافه کنید.

۳- کد زیر را در Form1 وارد کنید:

```
Private Sub Command1_Click()
   Dim cx As Long
   Dim cy As Long
   Dim RetVal As Long

   ' Determine if screen is already maximized.
```

```
        If Me.WindowState = vbMaximized Then
           ' Set window to normal size
           Me.WindowState = vbNormal
        End If

        ' Get full screen width.
        cx = GetSystemMetrics(SM_CXSCREEN)

        ' Get full screen height.
        cy = GetSystemMetrics(SM_CYSCREEN)

        ' Call API to set new size of window.
        RetVal = SetWindowPos(Me.hwnd, HWND_TOP, 0, 0, cx, cy, _
           SWP_SHOWWINDOW)
    End Sub
```

۴- یک ماژول جدید به پروژه اضافه کنید.

۵- کد زیر را در ماژول وارد کنید:

```
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

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

      Public Const SM_CXSCREEN = 0
      Public Const SM_CYSCREEN = 1
      Public Const HWND_TOP = 0
      Public Const SWP_SHOWWINDOW = &H40
```

۶- برنامه را اجرا کنید.

۷- روی CommandButton کلیک کنید. ملاحظه خواهید کرد که پنجره تمام صفحه را می پوشاند، از جمله روی تمام Taskbar ها را.

این روش در مورد تمام taskbar ها روی مونیتور اصلی عمل می کند، از جمله taskbar ویندوز و هر taskbar دیگری (مانند taskbar همراه Microsoft Ofiice) که از DATI[۲] استفاده کند.

[۲] Desktop Application Toolbar Interface