فصل ۷

API[۱] ویندوز

هدف‌های رفتاری: پس از آموزش این فصل هنرجو می‌تواند :

- مفهوم API ویندوز را شرح دهد.
- دلیل نیاز برنامه‌های شما به روال‌های ویندوز را بیان کنید.
- کتابخانه‌های پیوند دینامیکی (پرونده‌های DLL) را شرح دهد.
- نحوهٔ اتصال ویژوال بیسیک به روال‌های API ویندوز از طریق دستور Declare را توضیح دهد.

در این فصل، نحوهٔ دسترسی به روال‌های داخلی ویندوز شرح داده می‌شود. هرچند ویژوال بیسیک می‌تواند تقریباً همه کارهای مورد نیاز شما را انجام دهد، ولی در برخی از برنامه‌ها اگر بخواهید از ویژوال بیسیک برای انجام بعضی قابلیت‌ها استفاده کنید، برنامه‌نویسی مشکل می‌شود. خوشبختانه در چنین شرایطی روال‌هایی در ویندوز وجود دارند که درون پرونده‌های DLL ذخیره شده‌اند.

با استفاده از روال‌های ویندوز می‌توانید قدرت ویژوال بیسیک را در برنامه‌های خودتان افزایش دهید و از آن بخواهید کارهایی را انجام دهد که فقط ویندوز حق اجرای آن‌ها را دارد. در این فصل، نه تنها نحوهٔ دسترسی به این روال‌های ویندوز توضیح داده می‌شود بلکه تعدادی از این روال‌ها که می‌توانید با آن‌ها کار کنید، بررسی می‌شوند.

۱ـ۷ـ API ویندوز

API ویندوز مجموعه‌ای از روال‌هاست که در دسترس برنامه‌نویس می‌باشد. به عبارت دیگر، این روال‌های API دقیقاً مانند توابع داخلی خود ویژوال بیسیک عمل می‌کنند. وقتی لازم است تا از کد یک روال API استفاده کنید، برنامه ویژوال بیسیک آن روال را فراخوانی می‌کند. زمانی که API

۱ـ Application Programming Interface

ویندوز به پایان رسید، کنترل به برنامه برمی‌گردد و اجرای آن ادامه پیدا می‌کند.

API سرنام Application Programming Interface به معنای رابط برنامه‌نویسی کاربردی است و به مجموعه‌ای از روال‌های داخلی ویندوز اطلاق می‌شود که می‌توانید آن‌ها را از ویژوال بیسیک فراخوانی کنید.

تمام روال‌های API ویندوز در پرونده‌های خاصی با نام DLL ذخیره می‌شوند. چند هزار روال API وجود دارد که می‌توانید آن‌ها را به کار ببرید. این روال‌های API درون پرونده‌هایی وجود دارند که در پوشه‌های Windows و Windows\System ذخیره شده‌اند. هنگام نصب ویندوز، پرونده‌های DLL، هم نصب می‌شوند. بنابراین، به‌طور خودکار به این کتابخانه‌ها دسترسی دارید.

DLL سرنام Dynamic Link Library به معنای کتابخانه پیوند پویاست. پرونده‌های DLL در دسترس برنامه‌هایی که به زبان ویژوال بیسیک و زبان‌های دیگری (که از DLL پشتیبانی می‌کنند) نوشته شده‌اند، قرار دارد.

اکثر پرونده‌های DLL دارای پسوند DLL. یا EXE. هستند. هر برنامه‌ای که می‌نویسید، به پرونده‌های DLL ویندوز دسترسی دارد.

سه پرونده DLL که معمولاً به کار می‌روند، عبارتند از :

- **USER32.DLL:** شامل توابعی است که محیط ویندوز و رابط کاربر مثل مکان‌نماها، منوها و پنجره‌ها را کنترل می‌کنند.
- **GDI32. DLL:** شامل توابعی است که خروجی برنامه به صفحه نمایش و ابزارهای دیگر را کنترل می‌کنند.
- **KERNEL32. DLL:** شامل توابعی است که سخت‌افزار و رابط‌نرم افزار داخلی ویندوز را کنترل می‌کنند. اکثر روال‌های مربوط به حافظه، پرونده و پوشه درون KERNEL32.DLL قرار دارند.

این سه پرونده، اغلب روال‌ها یا توابع API را نگه می‌دارند. شما می‌توانید این توابع را در برنامه‌های ویژوال بیسیک خودتان فراخوانی کنید. با یک نگاه کوتاه به پوشه‌های Windows و Windows\System چند کتابخانه پیوند دینامیکی دیگر را نیز می‌بینید مثل

COMDLG.DLL و MAPI32.DLL و NETAPI32 و WINMM.DLL.

همچنان‌که مایکروسافت قابلیت‌هایی را به سیستم عامل اضافه می‌کند، پرونده‌های جدید DLL هم ظاهر می‌شوند.

هنگامی که برنامهٔ جدیدی را در سیستم خودتان نصب می‌کنید، این برنامه DLL خاص خودش را ارایه می‌کند. بنابراین، در طول زمان تعداد زیادی پرونده DLL روی سیستم خواهید داشت.

۷-۲- پرونده‌های DLL

اصطلاح پیوند پویا معنای خاصی برای برنامه‌نویسان دارد. وقتی گفته می‌شود که یک روال با یک برنامه پیوند دینامیکی دارد، بدین معناست که این روال (سابروتین یا تابع) تا قبل از کامپایل برنامه، به آن متصل نمی‌شود. این تابع فقط در زمان اجرا در دسترس می‌باشد. توابعی که درون پنجره کد می‌نویسید دارای پیوند استاتیکی می‌باشند یعنی هر وقت برنامه را کامپایل می‌کنید، این توابع با بقیه کد اصلی ترکیب می‌شوند. اما پرونده‌های DLL با برنامه ترکیب نمی‌شوند. برنامهٔ شما به این روال‌ها در زمان اجرا دسترسی دارد اما پرونده EXE برنامه شامل روال‌های واقعی DLL نمی‌باشد.

نکته: مزیت استفاده از روال‌های DLL آن است که چند برنامه اجرایی تحت ویندوز می‌توانند به یک روال از پرونده DLL دسترسی داشته باشند. علاوه بر این همه کاربران باید روال‌های استاندارد DLL را داشته باشند. از آن‌جایی که برای اجرای یک برنامه ویژوال بیسیک، وجود ویندوز ضروری است، پس پرونده‌های DLL در دسترس می‌باشند.

۷-۳- دستور Declare

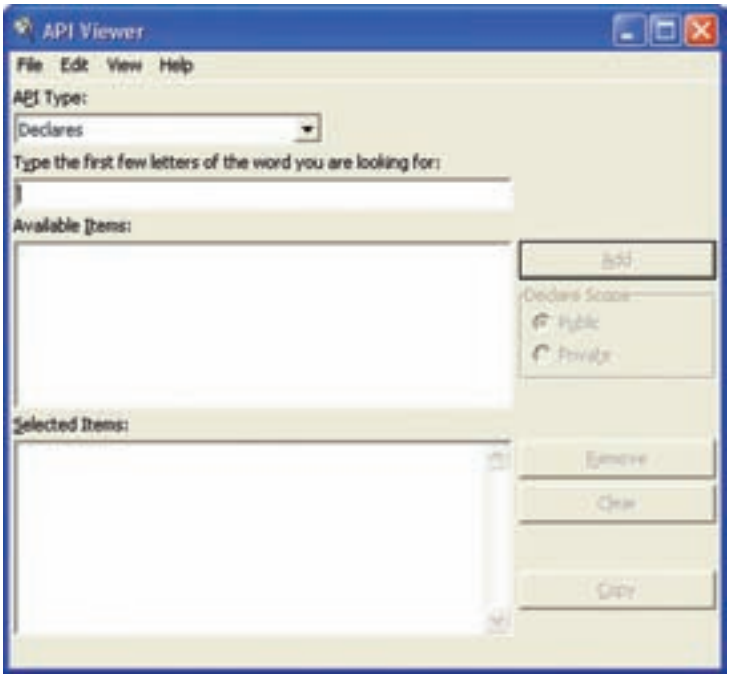
برای فراخوانی روال‌های API ویندوز باید از یک دستور خاص با نام Declare استفاده کنید. در مورد توابع داخلی ویژوال بیسیک به دستور Declare نیاز ندارید چون ویژوال بیسیک طرز کار توابع خودش و آرگومان‌های این توابع را می‌شناسد. اما روال‌های API خارج از میدان دید ویژوال بیسیک می‌باشند. بنابراین باید از این دستور استفاده کنید.

۱-۳-۷- ابزار API Viewer : ویندوز شامل هزاران روال API است که می‌توانید آن‌ها را فراخوانی کنید. حتی دانستن قالب تعداد کمی از این روال‌ها هم کار مشکلی است. در این رابطه، ویژوال بیسیک یک ابزار خاص به نام API Viewer دارد که برای راهنمایی در مورد قالب روال‌های API می‌توانید از این ابزار استفاده کنید.

ابزار API Viewer، روال‌های API را نمایش می‌دهد و آن‌ها را از نظر موضوع دسته‌بندی می‌کند طوری که می‌توانید روال‌های مورد نیاز خود را به سادگی پیدا کنید.

دکمهٔ Copy روی API Viewer اطلاعات مربوط به اعلان انتخاب شده را درون Clipboard ویندوز کپی می‌کند. همچنین اگر قبل از کلیک‌کردن روی دکمهٔ Copy، گزینه‌های Public یا Private این ابزار را کلیک کنید دیگر مجبور نیستید شناسه‌های اعلان را به صورت دستی تغییر دهید. زیرا API Viewer کلمات کلیدی مناسب را در دستور Declare مشخص می‌کند.

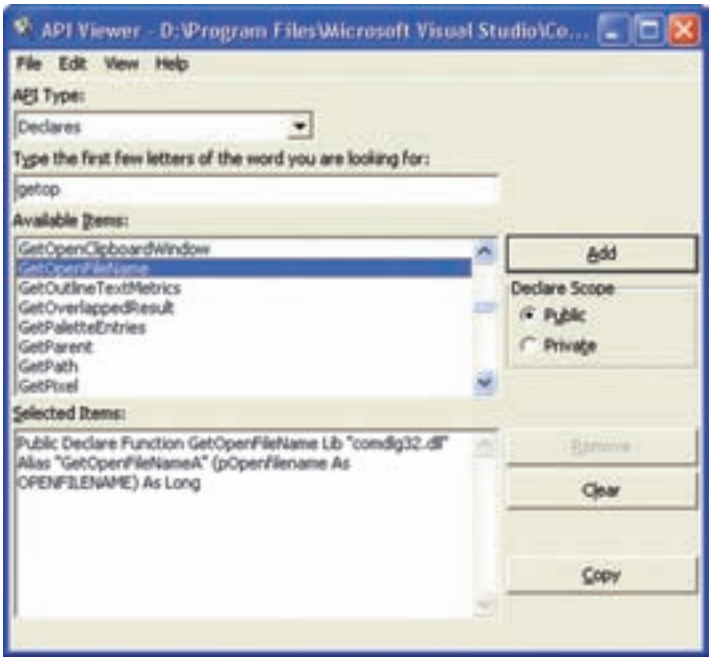
برای دسترسی به ابزار API Text Viewer ، از Start گزینه Programs، سپس گزینه Microsoft Visual Studio 6.0 و سپس گزینهٔ Microsoft Visual Studio 6.0 Tools و در پایان API Text Viewer را انتخاب کنید.



شکل ۱ــ۷ــ می‌توانید از طریق APIViewer به سادگی قالب روال‌های API را مشاهده کنید.

API Viewer اطلاعات اساسی خود را از پرونده‌های متنی APILOAD.txt، MAPI32.txt و WIN32API.txt پیدا می‌کند. این پرونده‌ها همراه API Viewer روی سیستم نصب می‌شوند.

از آن‌جایی که اکثر روال‌های API که موردنظر شما هستند، در پرونده WIN32API.txt قرار دارند. گزینه Load Text File را از منوی File پنجره API Viewer انتخاب و سپس پرونده WIN32API.txt را انتخاب کنید.



شکل ۲ــ۷

دقت کنید کادر لیست موجود در بالای پنجره دارای عنوان API Type است. با بازکردن این کادرلیست، سه مقدار زیر را مشاهده می‌کنید :

- **Constants:** همه ثابت‌های نامگذاری شده که پرونده API بارگذاری شده تشخیص می‌دهد را فهرست می‌کند.
- **Declares:** همه اعلان‌هایی که درون پرونده API بارگذاری شده ظاهر می‌شوند را فهرست می‌کند.
- **Types:** همه انواع داده‌ها که پرونده API بارگذاری شده تشخیص می‌دهد را فهرست می‌کند.

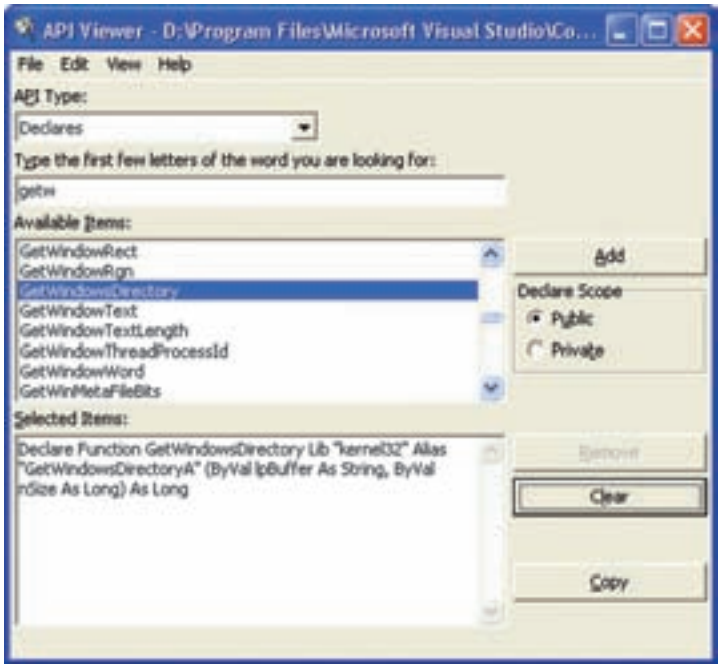
کادر لیست Available Items شامل کلیه روال‌های API ویندوز (مربوط به پرونده که بارگذاری کرده‌اید) و نوع مقداری است که می‌خواهید مشاهده کنید. مثلاً اگر می‌خواهید دستور Declare مورد نیاز روال GetWindowsDirectory را پیدا کنید، مراحل زیر را دنبال نمایید :

۱ــ از لیست API Type گزینه Declares را انتخاب کنید. چند دستور Declare درون لیست Available Items ظاهر می‌شوند.

۲ــ می‌توانید چند حرف اول یک دستور Declare خاص را درون کادر متن تایپ کنید تا این دستور به سرعت پیدا شود. بدین منظور Getw را تایپ کنید. همه گزینه‌هایی که با این حروف شروع می‌شوند درون کادر لیست Available Items ظاهر می‌شوند.

۳ــ این فهرست را مرور کنید تا گزینه GetWindowsDirectory را پیدا کنید.

۴ــ روی گزینه GetWindowsDirectory دابل کلیک کنید تا دستور Declare مورد نیاز تابع مطابق شکل ۳ــ۷ نمایش داده شود.



شکل ۳ــ۷ــ API Viewer دستور Declare مورد نیاز برای دستوری که انتخاب کرده‌اید را نمایش می‌دهد.

اکنون می‌توانید تمام دستور Declare را کپی کنید و آن را درون پنجره کد برنامه خودتان قرار دهید.

۴ـ۷ـ تعریف تابع API

قبل از این‌که کار با توابع را شروع کنیم بهتر است با فرم کلی توابع API آشنا شویم :

[Public| Private] Declare Function Function _ name Lib "DllFilename"_
[alias "Function alias"] (argument _ List) as data_type

دستور Declare برای تعریف تابع استفاده می‌شود. این دستور می‌تواند داخل یک ماژول یا فرم به کار رود. اگر داخل فرم استفاده شود کلید واژه Private برای آن به‌کار می‌رود و اگر داخل ماژول به کار رود، می‌تواند با یکی از کلید واژه‌های Public یا Private استفاده شود.

Function name، نام تابع API است که برای فراخوانی تابع مورد استفاده قرار می‌گیرد.

Dll–Filename، نام پرونده Dllای است که این تابع داخل آن قرار دارد. این نام نباید شامل مسیر باشد زیرا مسیر این پرونده‌ها ثابت است. نوشتن پسوند این پرونده الزامی نیست.

Function alias، این پارامتر اختیاری است. نامی است که داخل پرونده dll قرار دارد.

Argument List، لیستی از آرگومان‌های تابع می‌باشد. این لیست نشان می‌دهد چه تعداد متغیر و از چه نوعی باید به تابع فرستاده شود. نحوه تعریف آرگومان‌ها به شکل زیر است :

[byval¦ byRef] argument – Name as data – type

که در آن argument – Name نام آرگومان مورد نیاز است. انتخاب این نام اختیاری است و data – type نوع آرگومان را مشخص می‌کند.

کلید واژه ByVal، ByRef روش‌های خاص ارسال پارامتر به تابع API است. این دو روش کاملاً متمایز هستند. روش و متد ByRef به طور معمول استفاده می‌شود مگر اینکه به صراحت از ByVal استفاده شود. به همین دلیل شما در موقع تعریف API کلمه ByRef را نمی‌بینید و فقط به ByVal نیاز است.

ByVal به این مفهوم است که تابع API نتواند مقدار این متغیر را تغییر دهد. ولی ByRef به معنی «با مرجع» می‌باشد (By Refrence) و به این مفهوم است که تابع API می‌تواند مقدار این متغیر را تغییر دهد. با این روش نمی‌توان یک مقدار ثابت را ارسال کرد.

type ـ Data نوع داده‌ای است که تابع برمی‌گرداند.

۱ـ۴ـ۷ـ تابع Messagebeep : یکی از ساده‌ترین روال‌های API است. این تابع یکی از دو کار زیر را انجام می‌دهد :

اگر آرگومان ارسالی به این تابع مثبت باشد، صدای بوق از طریق کارت صوتی رایانه به گوش می‌رسد.

شکل کلی این تابع به صورت زیر است :

```
Private Declare Function messagebeep Lib 'user32' alias 'message ـ beep'
(byval wtype as long) as long
```

دستور Declare دقیقاً به ویژوال بیسیک اعلام می‌کند که چگونه تابع messagebeep را پیدا کند و چگونه مقادیر را منتقل نماید. این تابع درون پرونده user32.dll است.

مثال ۱-۷

باز و بسته شدن درب CD ROM : بر روی فرم دو دکمه فرمان برای OPEN و CLOSE قرار دهید. و تابع mciSendString را توسط APIViewer اضافه نمایید.

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA"
  (ByVal lpstrCommand As String, ByVal lpstrReturnString As String, _
  ByVal uReturnLength As Long, ByVal hwndCallback As Long) As
LongPrivate Sub CMDOPEN_Click()
  mciSendString "Set CDAudio Door Open Wait", 0&, 0&, 0&
End Sub
Private Sub CMDCLOSE_Click()
  mciSendString "Set CDAudio Door Closed Wait", 0&, 0&, 0&
End Sub
```

تحقیق : برنامه قبل را طوری تغییر دهید که بر اساس زمان بندی مشخصی درب CD ROM باز و بسته شود.

مثال ۲-۷

نمایش نام کاربر ویندوز : فرمی به همراه یک کادر متن ایجاد کنید و کد صفحهٔ بعد را بنویسید :

```
Private Declare Function GetUsername Lib "advapi32.dll" Alias
"GetUsernameA" (ByVal lpBuffer As String, nSize As Long) As Long

Private sub Form_Load()
  Dim Buffer As String

Buffer = String (255,0)
```

رشته‌ای به طول ۲۵۵ ایجاد کرده و آن را با کاراکتری با کد صفر پر می‌کنیم

```
GetUsername Buffer, 255          ' نام کاربر ویندوز را دریافت می‌کنیم
Buffer = Left$ (Buffer, InStr(Buffer, Chr$(0))-1)
                                 ' قسمتی از رشته را جدا می‌کنیم که نام کاربر در آن قرار دارد
Text1 . Text = Buffer            ' نام کاربر را در text box نمایش می‌دهیم
End Sub
```

در این کد به کمک تابع Instr. شمارهٔ اولین کاراکتر با کد صفر در رشتهٔ Buffer را پیدا کرده و به کمک تابع left$ قسمتی از رشته را که قبل از این کاراکتر قرار دارد (نام کاربر) جدا می‌کند.

مثال ۳-۷

تشخیص بزرگی و کوچکی و نوع (عددی ـ حرفی) نویسه‌های وارد شده از صفحه کلید:

یک فرم ایجاد کنید و کد زیر را در آن بنویسید :

```
Private Declare Function IsCharAlpha Lib "user32" Alias "IsCharAlphaA"
(ByVal cChar As Byte) As Long

Private Declare Function IsCharAlphaNumeric Lib "user32" Alias
"IsCharAlphaNumericA" (ByVal cChar As Byte) As Long

Private Declare Function IsCharLower Lib "user32" Alias "IsCharLowerA"
(ByVal cChar As Byte) As Long

Private Declare Function IsCharUpper Lib "user32" Alias "IsCharUpperA"
(ByVal cChar As Byte) As Long

Private Sub Form_KeyPress(KeyAscii As Integer)
    Dim Str As String
   Me.Cls
   If IsCharAlphaNumeric(KeyAscii) Then Str = "alphabetic Numeric"
```

```
If IsCharAlpha(KeyAscii) Then Str = “alphabetic”
If IsCharLower(KeyAscii) Then Str = Str + “Lower”
If IsCharUpper(KeyAscii) Then Str = Str + “Upper”
Me.Print “You pressed: ” + Chr$(KeyAscii)
Me.Print “This is: ” + Str
End Sub
```

در صورتی که کلید فشرده شده حرفی باشد خروجی تابع Is charAlpha مساوی True است و اگر حرفی یا عددی باشد خروجی تابع Is char Alpha Numeric مساوی True است و اگر کلید حرفی فشرده شده از حروف بزرگ باشد (مثل A) خروجی تابع Is char Upper مساوی True و اگر از حروف کوچک باشد مثل a خروجی تابع Is charlower مساوی True است.

مثال ۴-۷

مخفی یا ظاهر کردن Taskbar ویندوز:

بر روی فرم دو دکمه فرمان برای ظاهر و مخفی شدن نوار ایجاد نمایید و کدهای زیر را بنویسید(با استفاده از توابع SetWindowPos و FindWindow)

```
Private Hwnd1 As Long
Private Const SWP_HIDEWINDOW = &H80
Private Const SWP_SHOWWINDOW = &H40
Private Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long,
 ByVal hwndlnsertAfter 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 Declare Function FindWindow Lib "user32" Alias "FindWindowA"
(ByVal lpClassname As String,ByVal lpWindowName As String) As Long
Private Sub HideTask_Click()
   Hwnd1 = FindWindow("Shell_Traywnd", "")
   Call SetWindowPos(Hwnd1, 0, 0, 0, 0, 0, SWP_HIDEWINDOW)
End Sub
```

```
Private Sub ShowTask_Click()
   Call SetWindowPos(Hwnd1, 0, 0, 0, 0, 0, SWP_SHOWWINDOW)
End Sub
```

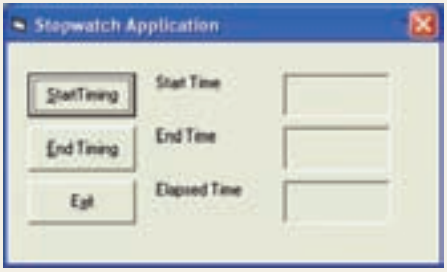
خودآزمایی

(**توضیح** :تمام تمرین‌های این فصل را با استفاده از توابع API بنویسید.)

۱ــ API چیست؟ کاربرد آن را شرح دهید.

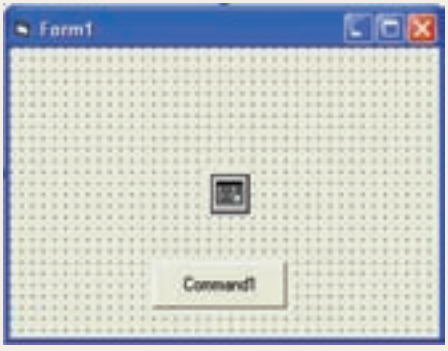
۲ــ انواع پرونده‌های DLL را نام ببرید و کاربرد هرکدام را توضیح دهید.

۳ــ فرمی به شکل زیر طراحی کرده و مشخصه‌های آن را تنظیم کنید. با کلیک روی دکمه‌های Start Timing و End Timing زمان سیستم خوانده شده و در برچسب‌های مقابل آن‌ها نمایش داده می‌شود. سپس اختلاف آن‌ها محاسبه و در برچسب سوم نمایش داده می‌شود.



شکل ۴ــ۷

۴ــ به وسیلهٔ شیء CommonDialog یک پرونده صوتی از نوع WAVE را انتخاب و به وسیله تابع SndPlaySound آن را پخش کنید.



شکل ۵ــ۷

۵ــ این مثال، تصویر زمینه دسک‌تاپ و همچنین تصویر کل صفحه نمایش را در یک شیء تصویر کپی می‌کند.

شکل ۶ــ۷

توضیح : در این تمرین از توابع API زیر استفاده کنید :

۱) GetDesktopWindow()

۲) GetDC()

۳) PaintDesktop()

۴) ReleaseDC()

۵) SearchBlt()