VISUAL BASICS

by Barry Seymour
March 8th, 1992

Reading and Writing INI Files

The Windows API provides a quick way for your program to work with INI files, the text files which contain operating parameters for many Windows programs. The API calls GetPrivateProfileString and WritePrivateProfileString make it possible for your application to read and write INI files following the standard Windows formats for these files. Without these functions you'd have to write hundreds of lines of code to read, parse and/or write the text files yourself.

The INI file format requires three parameters; a section name, a key name and a value, in the following format:

```
[Section]
Key=Value
```

WIN.INI follows this format. If you look at WIN.INI with a text editor, the first three lines are similar to...

```
[windows]
load=calendar.exe
run=write.exe
```

'Windows' is the section name, the keys are 'load' and 'run,' and the values are 'calendar.exe' and 'write.exe.' As many users know, the load and run commands can be used to specify which programs start when Windows does; programs 'loaded' are run minimized, where programs 'run' are normal or maximized and are active upon startup.

Function Declarations

Here is how the two API calls should be declared in your form or module:

```
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal SectionName
As String, ByVal KeyName As String, ByVal Default As String, ByVal
ReturnedString As String, ByVal MaxSize, ByVal FileName As String)

Declare Function WritePrivateProfileString Lib "Kernel" (ByVal
SectionName As String, ByVal KeyName As String, ByVal NewString As
String, ByVal FileName As String)
```

Note, as always, that each function declaration should be on one line.

GetPrivateProfileString / StringFromINI

GetPrivateProfileString is a function that retrieves a string from an INI file. In calling the function, you need to pass the section and key names, the default value to return if the read fails and the name of the INI file.

The GetPrivateProfileString function requires some preparation before you can call it. As with GetWindowsDirectory, a string variable has to be defined and prepared to receive the return value. In addition, a default value must also be established in the event that the read fails or there is a null string returned. The wrapper function StringFromINI performs this preparation.

Reading an INI File

```
Function StringFromINI (SectionName As String, KeyName As String,
Default As String, FileName As String) As String

'ALL OF THESE PARAMETERS MUST BE INITIALIZED for this API call to work.

   MaxStringLen% = 255 'set string length
   ReturnedStr$ = Space$(MaxStringLen%) ' init the receiving string

   Result% = GetPrivateProfileString(SectionName, KeyName, Default,
   ReturnedStr$, MaxStringLen%, FileName) ' ALL ON ONE LINE!

   ResultStr$ = Left$(ResultStr$, Len(ResultStr$) - 1) ' REMOVE CHR$(0)
                                                       ' FROM END
   ResultStr$ = LTrim$(RTrim$(ReturnedStr$)) 'TRIM OUT BLANKS
   StringFromINI = ResultStr$
End Function
```

As you can see, the receiving string must be initialized to a specific length, and that string plus the value of the length must be passed to the API call. Upon return, the trailing null character (Chr$(0)) and any remaining blank spaces are removed. This way just the resulting value is returned to the calling procedure in a form Visual Basic can handle.

WritePrivateProfileString / StringToINI

Writing information to an INI file is much simpler. Simply load the variables and make the function call WritePrivateProfileString. Although a wrapper really isn't needed, the function StringToINI is suggested for consistency's sake.

Writing an INI file...

```
Function StringToINI (SectionName As String, KeyName As String,
DataString As String, INIFileName As String) As Integer
   StringToINI=WritePrivateProfileString(SectionName, KeyName,
DataString, INIFileName)
End Function
```

This function returns an integer result. If the result is zero, the function failed; if nonzero, it succeeded.

Uses of INI files

INI files can be used to store any information required by a program; default file extensions, the location of supplementary files, program operating parameters or defaults, the names of the last four files edited and much more. This information can be read at program startup or any time thereafter; changes to those parameters can be written to disk instantly.

In many cases this eliminates the need for global variables; simply read the desired information from the INI file! Look through the INI files cre ated by the Windows applications you're currently running and you'll see excellent examples of what kind of information is stored there.

The example file for this week's column is VBEX3.ZIP. If you download it you will see all of these examples in action, plus the use of the GetWindowsDirectory and WinDir functions discussed last week.

Next Week...

Next week we'll look at a few more API calls, and how they can be used to obtain more information about the environment your program will be running in. The wrapper functions GetWinInfo and EXEName will be demonstrated, showing how you can determine Windows' operating mode and available memory and how you can get the name and location of the EXE file your program is running from.

Have fun!

Barry Seymour
Marquette Computer Consultants
San Rafael CA 415/459-0835