Calling the Windows API from VPascal

27 June 2008

This document describes how to directly call functions in the Windows API from your own VPascal modules.

Introduction

Using VPascal, you almost never need to worry about Windows. But occasionally, a module may have specialized requirements that require it to call Windows API functions directly. You can do so using VPascal's ability to call external DLLs.

The Windows API interface is implemented as a set of DLLs (like KERNEL32.DLL, USER32.DLL etc) that export the functions programmers need to communicate with the operating system. If we know the name and parameter list for one of these function we can link to it in VPascal and call it when the module is running.

Linking to DLLs

Let's start by reviewing how to call a function in an external DLL from VPascal.

The DLL concerned can be written in any language but must use the Windows standard calling model for passing parameters. We need to know the name of the DLL file, the exported name of the function we wish to call and the number, order and types of the parameters to pass to the function.

Declaring an external function is the only time in VPascal that you have to use explicit data types. Seven data types are recognized for this purpose: `byte`, `shortint`, `word`, `longint`, `single`, `double` and `pointer`. The `integer` type is also recognized and in V++ is equivalent to `longint`. At runtime, VPascal automatically converts the parameters you pass to an external function to the declared types.

You can't declare new types but many Windows types can be declared as one of the accepted types. For example, a Windows handle can be declared as a `longint` (which is what it actually is).

Windows API functions

Consider the Windows SendMessage function which is used to send Windows messages to applications or to individual windows. SendMessage is exported by the USER32.DLL library and can be declared in VPascal as follows:

```
function SendMessage( Handle,Message,wParam,lParam:integer ) : integer ;
external 'User32' ; name 'SendMessageA' ;
```

SendMessage is declared above as a function but it's also possible to declare it as a procedure if you don't need the result. Note also that we used the **name** keyword to specify the actual name used in USER32.DLL to export the function.

Having declared it, you can now call SendMessage just as you would call any other VPascal function.

See the next page for a practical example.

PO Box 35-715 • Browns Bay • Auckland 1330 • New Zealand
Tel +64 (9) 478 5779 • Fax +64 (9) 479 4750
info@digitaloptics.co.nz • www.digitaloptics.co.nz

Example
As a practical example, consider how to shut down V++ from a module (there's no built-in function to do this). You can shut down by using SendMessage to send the main V++ window a Windows wm_Close message.

In order to send the message, you first need to declare another API function, FindWindow, which you use to determine the handle of the main V++ window.

The technique is illustrated in the following sample module:

```
Program Close ;
{ Sample module that shuts down V++ by calling the Windows API }

function FindWindow( ClassName,WindowName:pointer ) : integer ;
external 'User32' ; name 'FindWindowA' ;

procedure SendMessage( Handle,Message,wParam,lParam:integer ) ;
external 'User32' ; name 'SendMessageA' ;

const
  wm_Close = 16 ;

begin
  SendMessage( FindWindow( 'TFrameForm','V++' ),wm_Close,0,0 ) ;
end.
```

Note how the FindMessage is used to get a main window handle for V++.

Other messages can be sent in exactly the same way. Please refer to the Windows documentation for information about the available messages and their parameters.

For more information contact:

Digital Optics Ltd
PO Box 35-715
Browns Bay
Auckland 0753
NEW ZEALAND

E-mail: info@digitaloptics.co.nz
Web: www.digitaloptics.co.nz

Phone: +64 (9) 478 5779
Fax: +64 (9) 479 4750