



# maXbox Starter 8

## Start with Operating System Programming

### 1.1 Admin your System

Today we spend another time in programming with your operating system (called OS) and the API. An API (Application Programming Interface) is the port to the library of your system. Most of the VCL is designed for working with the Windows API or the CLib of Linux. This is handled in a way analogous to the way a call to a DLL (dynamic link librarie) works: The VCL<sup>1</sup> is a managed API that calls into the Win API, marshalling between the managed structures on the VCL side and the unmanaged types and operations that the Win API uses. Hope you did already work with the Starter 1 to 7 to get more training available at:

<http://www.softwareschule.ch/maxbox.htm>

This lesson will introduce you to various short functions interacting with the OS. When your application calls a Win API, it is making a call into an (unmanaged) DLL. Because of this, all parameter values must be known before, where the OS can work with it, and results are then send back into memory. Also the CPU on which the OS relies must be known because of generic types. The generic integer types are Integer and Cardinal; use these whenever possible, since they result in the best performance for the underlying CPU and operating system.

When writing cross-platform applications, remember that although the Delphi language is not case sensitive, the Linux operating system is. When using objects and routines that work with files, be attentive to the case of file names.

Operating Systems today have become so complex that they require large numbers of developers, assembler programmer, testers and managerial overhead to develop. They rival large commercial enterprise application for smart phones or desktop system in their complexity and cost many millions of dollars to develop and market.

Let's begin with the call of an API function:

1. Search the name of the API (`user32.dll`)
2. Initialize resources or state and check parameters.
3. Locate the name of the function (`MessageBeep`)
  1. Declare the function (line 18).
  2. External the function (line 19).
4. Call the function in your app (`MessageBeep(sound)`)

The final ingredient of a call is to find the right parameters. No one wants to set the wrong parameter and type and most of the DLL's aren't documented well. So what's a DLL? Dynamic link libraries (DLLs) are modules of compiled code that work in conjunction with an executable or an app to provide


---

<sup>1</sup> VisualComponentLibrary

functionality to an application. You can create DLLs in cross-platform programs. However, on Linux, DLLs (and packages) recompile as shared objects.

## 1.2 Code the System

As you already know the tool is split up into the toolbar across the top, the editor or code part in the centre and the output window at the bottom.

 Before this starter code will work you will need to download maXbox from the website. It can be downloaded from <http://sourceforge.net/projects/maxbox> site. Once the download has finished, unzip the file, making sure that you preserve the folder structure as it is. If you double-click `maxbox3.exe` the box opens a default program. Test it with F9 or press **Compile** and you should hear sound. So far so good now we'll open now the example in the folder `/examples`:

`192_opearatingsystem2.txt`

If you can't find the file use the link:

[http://www.softwareschule.ch/examples/192\\_opearatingsystem2.txt](http://www.softwareschule.ch/examples/192_opearatingsystem2.txt)

Or you use the `Save Page as...` function of your browser<sup>2</sup> and load it from `examples` (or wherever you stored it). Now let's take a look at the code of this project. Our first line is

```
12 program OperatingSystem_Tutorial_Delphi;
```

We have to name the program it's called like above.


```
17 // for DLL Load Demo
18 function MessageBeep(para: integer): byte;
19     external 'MessageBeep@user32.dll stdcall';
20
21 const RESPATH = '\\SYSTEM\\CurrentControlSet\\services\\SCSService';
22     REGROOT = HKEY_LOCAL_MACHINE;
23
24     REGPATH3 = '\\Software\\Microsoft\\Internet Explorer\\TypedURLs';
25     REGROOT3 = HKEY_CURRENT_USER;
```

When creating a DLL that uses the VCL or CLX, the required VCL or CLX components are linked into the DLL resulting in a certain amount of overhead. The impact of this overhead on the overall size of the application can be minimized by combining several components into one DLL that only needs one copy of the VCL and CLX support components.

In our case we don't create a DLL we just call it. First in line 18 we declare the function and in line 19 we declare it as external. Note the external keyword is required to find the DLL in a well known path. Be sure the file exists and the path is one of the environment variables. Otherwise you put the DLL in the local path of the maXbox exe. In line 21 you find a declaration we use later in this script.

The exported functions of that DLL then become available for use by your app. Prototype the DLL functions as below, so the function name can be an alias, e.g. `myMessageBeep`:

```
18 function myMessageBeep(para: integer): byte;
19     external 'MessageBeep@user32.dll stdcall';
```

 You can also load a DLL at runtime by calling the `LoadLibrary` function. The example requires many objects of the classes: `TForm`, `TStringGrid`, `TFileStream`, `TPrinter`, `TIdHTTP`, `TSpeedButton`, `TComponent`, `TApplication`, `TGraphic`, `TPicture` and `TBitmap` and the bitmap passes your images with the help of a `LoadFromFile(BITMAPPATH)` to the screen.

---

<sup>2</sup> Or copy & paste

Be sure this relative path `BITMAPPATH` points to the bitmap. That needs an explanation: For Win API functions that manipulate files, file names can often be relative to the current directory, while some APIs require a fully qualified path. A file name is relative to the current directory if it does not begin with one of the following:

- A UNC name of any format, which always starts with two backslash, characters ("`\\`"). UNC is mostly used on a file server.
- A disk designator with a backslash, for example "`C:\`" or "`d:\`".

So we use a relative path:

```
28 BITMAPPATH = '..\maxbox3\examples\citymax.bmp';
```

Relative paths can combine both example types, for example "`C:..\tmp.txt`". This is useful because, although the system keeps track of the current drive along with the current directory of that drive, it also keeps track of the current directories in each of the different drive letters.

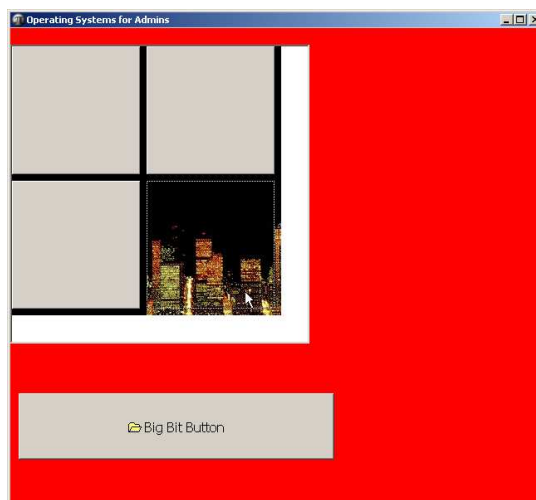
Let's change the topic and call now your DLL function; that's how we call it:

```
39 if sound mod 10 = 0 then begin
40     MessageBeep(sound)
41     Sleep(1500);
42 end
```

This routine can be found in our function `MessageBeep` which is called by the `Signal_Sounds` procedure, so we jump now to the next function `getFileCount`. The function counts all files in your root or exe folder.

```
54 Function getFileCount: integer;
```

There are three routines used for finding a file: `FindFirst`, `FindNext`, and `FindClose`. `FindFirst` searches for the first instance of a filename with a given set of attributes in a specified directory. `FindNext` returns the next entry matching the name and attributes specified in a previous call to `FindFirst`. `FindClose` releases memory allocated by `FindFirst`. You should always use `Close` to terminate a `First/Next` sequence. If you want to know if a file exists, a `FileExists` function returns `True` if the file exists, `False` otherwise.



1: The Big Bit Button calls a resource

`FindFirst` is very versatile.

```
79 Found:= FindFirst(Folder, faDirectory); //like *.*
```

A `TSearchRec` defines the file information searched for by `FindFirst` or `FindNext`. Now we have a Windows application that we can run, but the form doesn't show the result. The form has no other controls on it (beside a button and a grid) to display any information. It's up to you! In a regular Win Forms application we would now add such controls like a list to the form to build our final app, but for

our tutorial we are going to write out everything using the `WriteLn` procedure. In line 274 and 275 you see the output. Next I'll show you another output by `ShowMessage`. The `ShowMessage` procedure displays a string of text in a simple dialog with an OK button.

```
277   if GetTextFromFile(ExePath+'firstdemo3.txt',rStr) then ShowMessage(rStr);
```

The function uses a `FileStream` to open the file. Because `TFileStream` is a stream object, it shares the common stream methods. You can use these methods to read from or write to the file, copy data to or from other stream classes, and read or write components values.



A note about the upper functions with filenames. In cross-platform applications, you should replace any hard-coded pathnames with a correct path for the system or use environment variables.

On line 113 we get the `DriveTypes`. `DiskSize` returns -1 if the drive number is invalid. Drive parameter can be set to: 0 = Current drive, 1 = A, 2 = B, 3 = C and so on. At the end of the code you see the specification of the function `GetDriveType`.

```
case GetDriveType(PChar(chr(Drive) + ':\')) of
  0: //DRIVE_UNKNOWN
    Memo2.Lines.Add(DriveLetter + ' Unknown Drive');
  {1: //DRIVE_NO_ROOT_DIR
    Memo2.Lines.Add(DriveLetter + ' No Root Dir or mount Point');}
  2: //DRIVE_REMOVABLE:
    Memo2.Lines.Add(DriveLetter + ' Removable or Floppy Drive');
```

For example, The `GetDriveType` API function determines whether a disk drive is a removable, fixed, CD-ROM, RAM disk, or network drive. The procedure lists all the drives and their types on a users computer.



Place a button and another memo component on our form and assign an `OnClick` handler of a button around the procedure in line 113.

Next function is a prototype of a timecounter in line 139.

```
time:= GetTickCount;
maxForm1.StatusBar1.SimpleText:= ('Time Diff: '+IntToStr(GetTickCount-time));
```

The `GetTickCount` gives us the number of milliseconds since Win was started. The accuracy of this API, however, is only 55 ms, as it is updated 18.2 times each second (just like the real mode system clock). These timestamps use the millisecond-resolution "time since Win started" returned by the `GetTickCount` function. Apps must be aware that this could be the tick count on the server (or machine where the service provider directly managing the hardware is running), and is not necessarily the same machine on which the application is executing; thus, the timestamps in these API messages can only be compared to each other, and not to the value returned by `GetTickCount` on the processor on which the application is running.

On line 149 and 157 we go back to graphics and bitmaps in an OS. The `BITMAP` resource-definition statement specifies a bitmap that an application uses in its screen display or as an item in a menu or control. A bitmap is a rectangular array of bits.

How graphic images appear in your application depends on the type of object whose canvas you draw on. If you are drawing directly onto the canvas of a control like our `StringGrid`, the picture is displayed immediately. However, if you draw on an off-screen image such as a `TBitmap` canvas, the image is not displayed until a control copies from the bitmap onto the control's canvas.

```
161   StringGrid1.Canvas.Draw(Rect.Left,Rect.Top, getBitmap);
```

That is, when drawing bitmaps and assigning them to an image control, the image appears only when the control has an opportunity to process its `OnPaint` message (VCL applications) or event (CLX applications).

A real interesting point is now to get some bitmaps, icons or other resources out from the box, I mean out from the maXbox!

```
195 with myButton do begin
196     Parent := frmMon;
.....
203     Glyph.LoadFromResourceName(getHINSTANCE, 'OPENFOLDER');
```

In addition to form files, each project uses resources to hold the application's icon and other resources such as strings or bitmaps. And such a bitmap we get with LoadFromResourceName. This procedure loads the specified bitmap resource along with palette information from a module's executable file. The instance is the handle of the module that contains the resource and ResName is the name of the resource to load.

**procedure** LoadFromResourceName(Instance: THandle; const ResName: String);

So the folder bitmap you see on the speedbutton is extracted from the exe inside, in our way from the maXbox3.exe itself.



Try it with another resource called HARD, FLOPPY or EXECUTABLE.

That's the reason I made two lines more with an example in case you'll need that.

 Big Bit Button

CopyRect: Copies part of an image from another canvas into the canvas and Draw renders the graphic object specified by the Graphic parameter on the canvas at the location given by the coordinates (X, Y).

### 1.3 OS Internet Resources

On line 223 we use the mighty function GetHostByName from the winsock.dll. The Win Sockets GetHostByName function gets host information corresponding to a hostname or domainname. A socket function on the other side creates a socket which is bound to a specific service provider. So this function is just a wrapper around some API commands like WSASStartUp. The WSASStartUp function initiates use of the Win Sockets DLL by a process.

```
IF WSASStartUp($101, WSA) = 0 THEN BEGIN
    GetMem(P, 255 + 1);
    StrPCopy(P, ComputerName);
    H := GetHostByName(P);
```



This function must be the first Win Sockets function called by an application or DLL. It allows an application or DLL to specify the version of Sockets required and to retrieve details of the specific Win Sockets implementation. After all we get an IP-Address back. On line 228 we found a second net function.

```
function GetURLByIndy(const AURL: String): String;
```

The Get method shows the HTML of the site. The Get and Put methods are overload methods, so in maXbox you have to name it with Get1 and Get2 and so on or try it with the standard method Get like in our script.

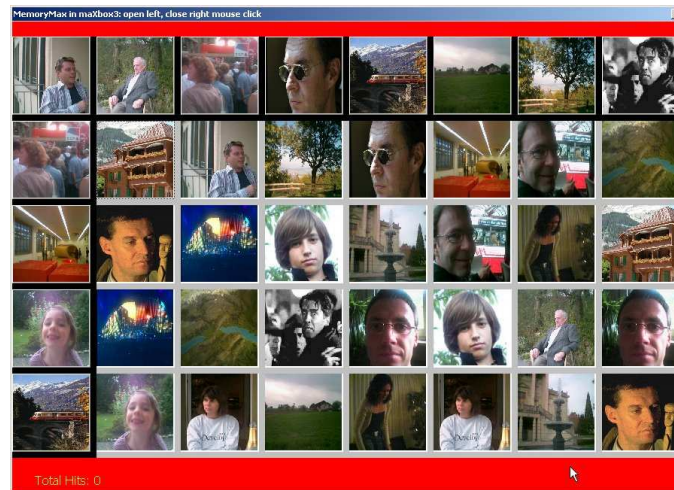
The acronym HTML stands for Hyper Text Markup Language - the primary programming language used to write content on the web. One of a practical way to learn more about actually writing HTML is to get in the maXbox editor and load or open a web-file with the extension .htm or html. Or you copy the output and paste it in a new maXbox instance. Then you click on the right mouse click (context menu) and change to HTML Syntax!

Line 245 and the call in line 288 get some insights in the registry of a Win OS. Under 32-bit Windows many configuration information is stored in the Win registry instead of .ini files, as was the case under

16-bit Windows. In our example we get some information about LocalMachine or CurrentUser therefore we need a KeyRoot, KeyPath and a Field to look for.

```
WriteLn('Registry read of url2: '+RegRead(REGROOT3, REGPATH3, 'url2'))
```

☞ Linux does not use a registry to store configuration information. Instead, you use text configuration files and environment variables rather than the registry. System configuration files on Linux are often located in /etc, such as /etc/hosts. Other user profiles are located in hidden files (preceded with a dot), such as .bashrc, which holds bash shell settings or .XDefaults, which is used to set defaults for X programs. Making the correct entries and deletions for install and uninstall is therefore a complex task.



## 2: Resources of Bitmaps

Let's back to a window and the well known **Handle**. An important concept in Win programming is the concept of an object (not in the OO way) handle. Many functions return a handle to an object that the function created or loaded from a resource. Internally, Win keeps track of all of these handles, and the handle serves as the link through the operating system between the object and the application. There are several ways to obtain the handle of a window in our way with the function FindWindow

```
253 Handle:= FindWindow(MAINFORM, ''); //FindReplDialog does not
```

The point is to get a handle <> 0 to prove, our application is alive.

☞ Since TCanvas is a wrapper resource manager around the Windows device context, you can also use all Win GDI functions on the canvas. The Handle property of the canvas is the device context Handle.

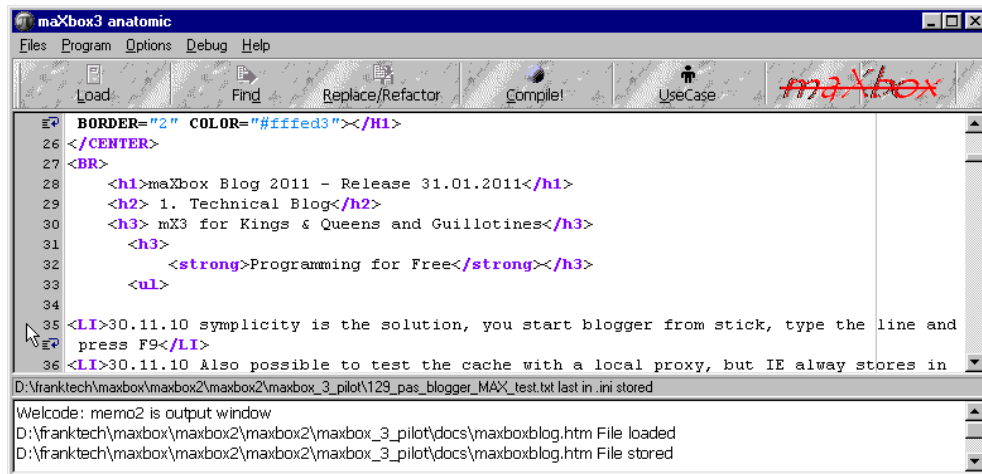
On the end we want to print a bitmap on line 260.

```
261 begin
    //initialization
265 Printer.Canvas.Draw((Printer.PageWidth - aGraphic.Width ) div 2,
266     (Printer.PageHeight - aGraphic.Height) div 2, aGraphic);
267 //Printer.Canvas.Draw(40,40, aGraphic);
268 Printer.Canvas.TextOut(480,480, 'Place any text here'); contentLst:=
```

In Delphi, you print via the TPrinter object.

- Add printers to your uses clause
- Use the Printer function to access the global instance of TPrinter
- Printer.BeginDoc starts the print job
- Printer.EndDoc stops the print job and sends it to the printer
- Printer.NewPage forces a new page
- Printer.Canvas is used to generate the output page

In the Main Section you find some more OS calls like the CurrentProcessID or NumberOfProcessors or the PageSize to discover on your OS.



Most of those functions after line 272 we show the new content or the result with `writeln` in the output window below the status line.



Try to uncomment the line 304 and start a script within a script!:

```
SHELL OPENER =  
  ' ExecuteCommand('cmd','/k dir *.*')
```

Some notes at last about console applications. Console applications are 32-bit programs that run without a graphical interface, in a console window. These applications typically don't require much user input and perform a limited set of functions.

Users can terminate console applications in one of the following ways:

Click the Close (X) button.

- Press Ctrl+C.
- Press Ctrl+Break.
- Log off.

I made an example of a console application in a simple way to execute:

[http://sourceforge.net/projects/maxbox/files/maXbox\\_Archive/wikimaxboxtest.zip/download](http://sourceforge.net/projects/maxbox/files/maXbox_Archive/wikimaxboxtest.zip/download)

[max@kleiner.com](mailto:max@kleiner.com)

Links of maXbox and a simple Web Server:

<http://www.softwareschule.ch/maxbox.htm>

<http://www.softwareschule.ch/download/httpserver2.zip>

<http://sourceforge.net/projects/maxbox>

<http://sourceforge.net/apps/mediawiki/maxbox/>