

Accessing MSVBVM60 API from VB IDE.

I'm writing this more as an informative thing than a real useful technique. I'm gonna try to explain how to patch/mod vb files in order to have access to any API exported by MSVBVM60 without declaring or adding anything else to your project.

The tools I'm going to be using to do this are:

1- PE Explorer

PE Explorer is the most feature-packed program for inspecting the inner workings of your own software, and more importantly, third party Windows applications and libraries for which you do not have source code.

Link : <http://www.heaventools.com/>

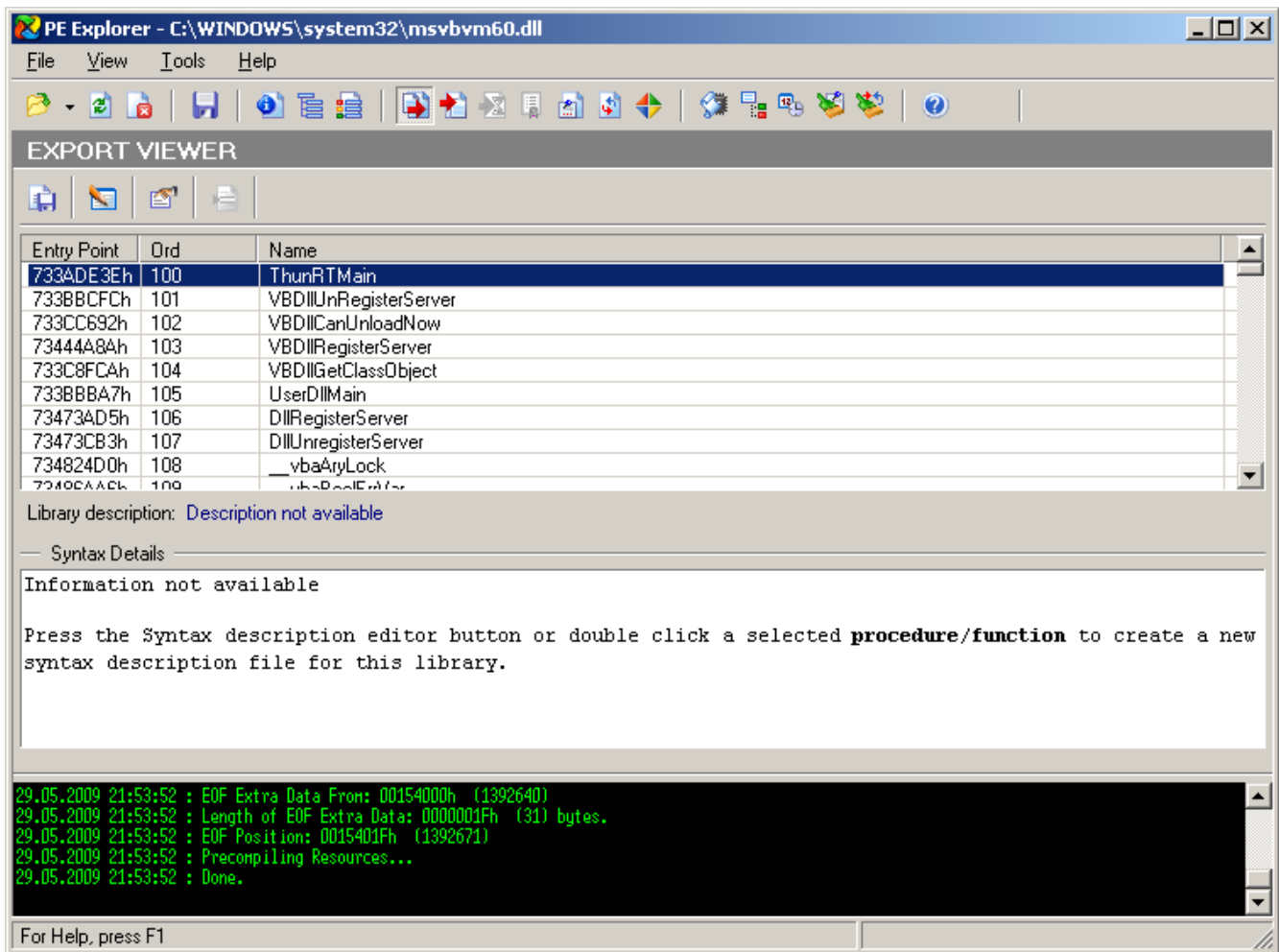
2- Type Library Editor

The PowerVB Type Library Editor control is used in all of the type library editing tools provided with Advanced Visual Basic 6: Power Techniques for Everyday Programs. Editing elements in a type library is essentially the same regardless of the exe/add-in that the control is hosted in. The editing operations fall into two categories, those done by editing fields in the right-hand side of the control, and commands executed from the context menu in the tree control. This document explains the context-menu commands for each type of selection.

Link : <http://www.powervb.com/>

Lets start with this. The first thing we have to do is find MSVBVM60.dll (located in the sytem32 folder) and see what functions exports. To do this open PE Explorer, File>Open> navigate to the system32 folder and select MSVBVM60.dll, once we have the file open we go to view>export or Ctrl+E.

You should see a list with all the exported functions an their ordinals. Some of this functions are commonly used in vb and well documented, but most of them are a mystery. You can Google for info about them.



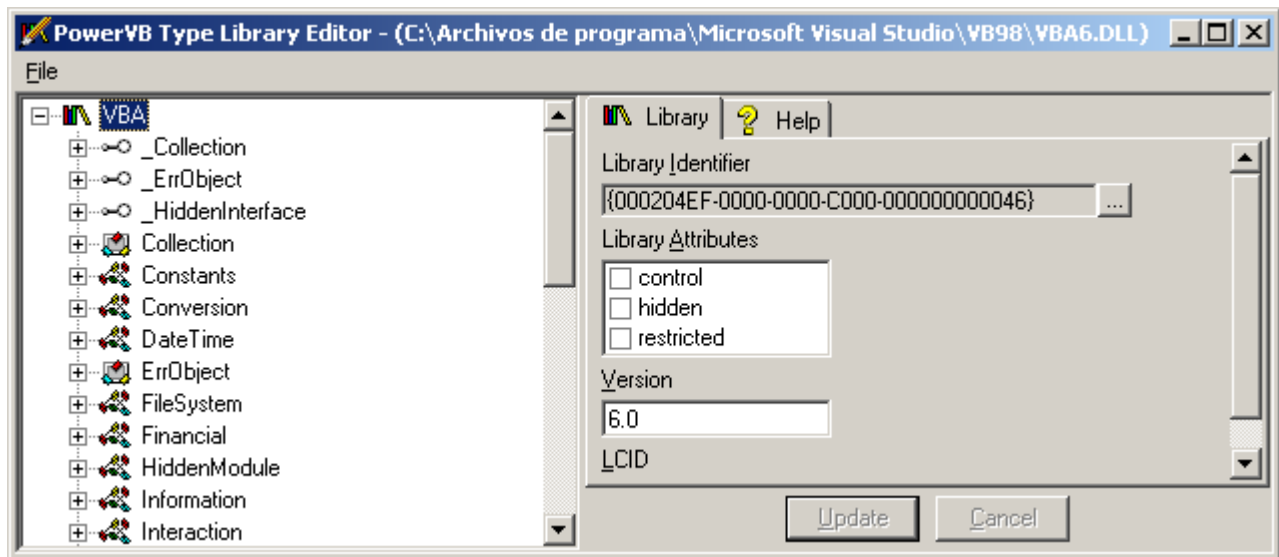
Lets save this info for future use. Click on save syntax details, navigate to your desktop and create a folder called “MSVBVM_Test” and save the text file there. Now we can close PE Exp.

Before start modding our VB, I highly recommend to install the SP6 because if you install this patch later it will overwrite all our work.

Time to start doing funny stuff. First navigate to you VB folder, find a file called VBA6.DLL and make a backup of it, just in case something goes wrong or we want to restore our VB.

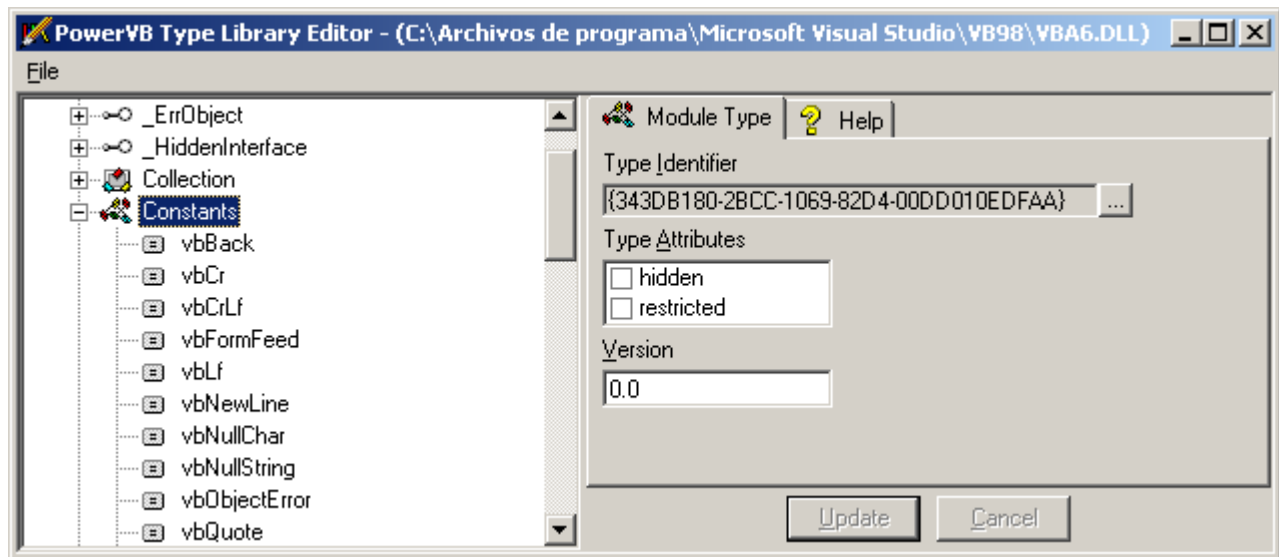
Lets start with a simple change, close all VB instances that might be running, now open Type Library Editor, File>Open> navigate to the VB folder and select VBA6.DLL.

You should see something like this



Now lets save the TLB in the folder we created previously in the desktop “MSVBVM_Test” name the file “VBA6.tlb”.

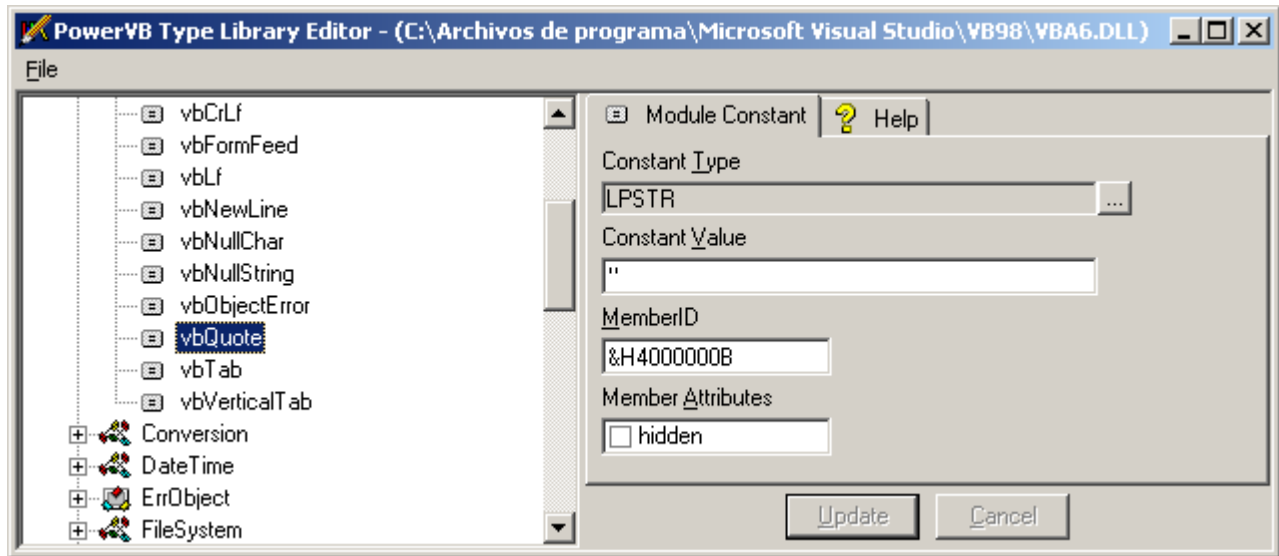
For the first mod we gonna do something simple in order to understand how it works. We gonna add a new constant to VB called vbQuote this new constant will represent a “ , in that way we can use vbQuote instead of chr\$(34) or “”””.



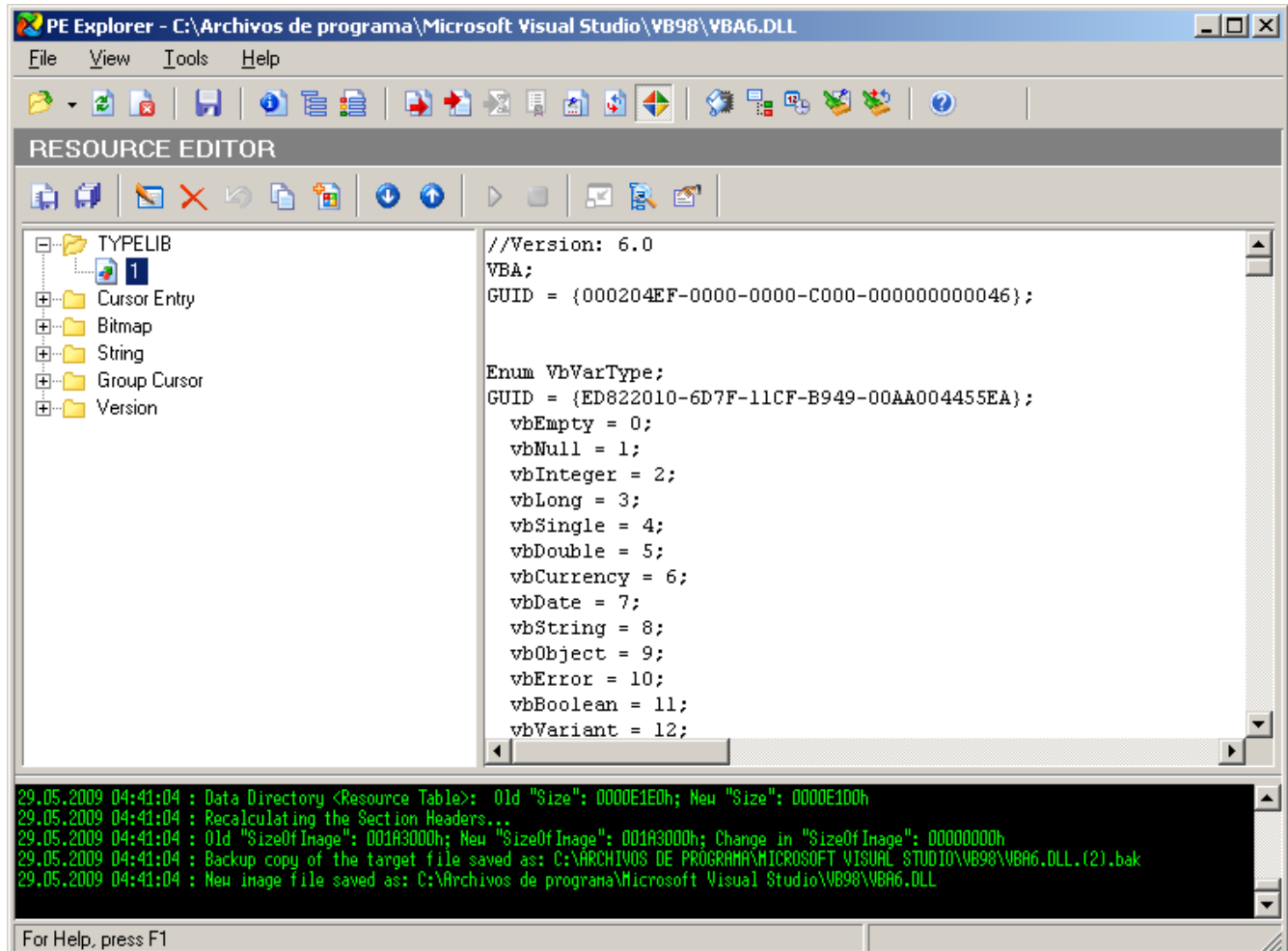
Go to the left had side tree-view and find a module called Constants, Expand that module and you will see a bunch of known VB constants (vbBack,vbCr, etc). Right Click on the constant module and click “Add Constant” this will add a new constant called Constant1.

Now select that constant Right Click > Rename and type vbQuote. On the right panel choose constant type “LPSTR” and in the “Constant Value field” type “ , click update and save the file.

It should look like this



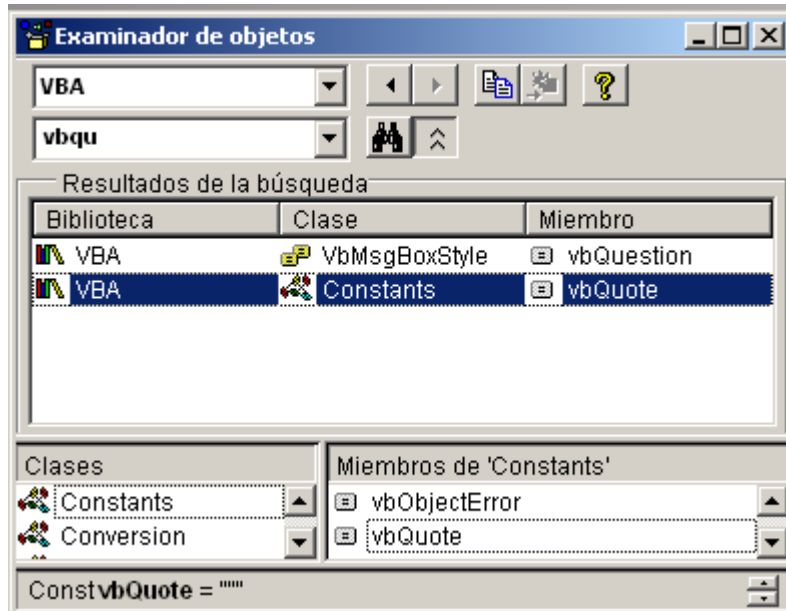
Is time to update the VBA6.DLL file to reflect the changes. Open PE Explorer File>Open> navigate to the VB folder and select "VBA6.DLL" now open the resource view, View>Resources or Ctrl+R



Select the first Type Library.

Click Resource Editor

Navigate to the TLB file located in the folder we created previously and replace the resource with the new file. Close PE Explorer and run your VB, now you should have a new constant available.



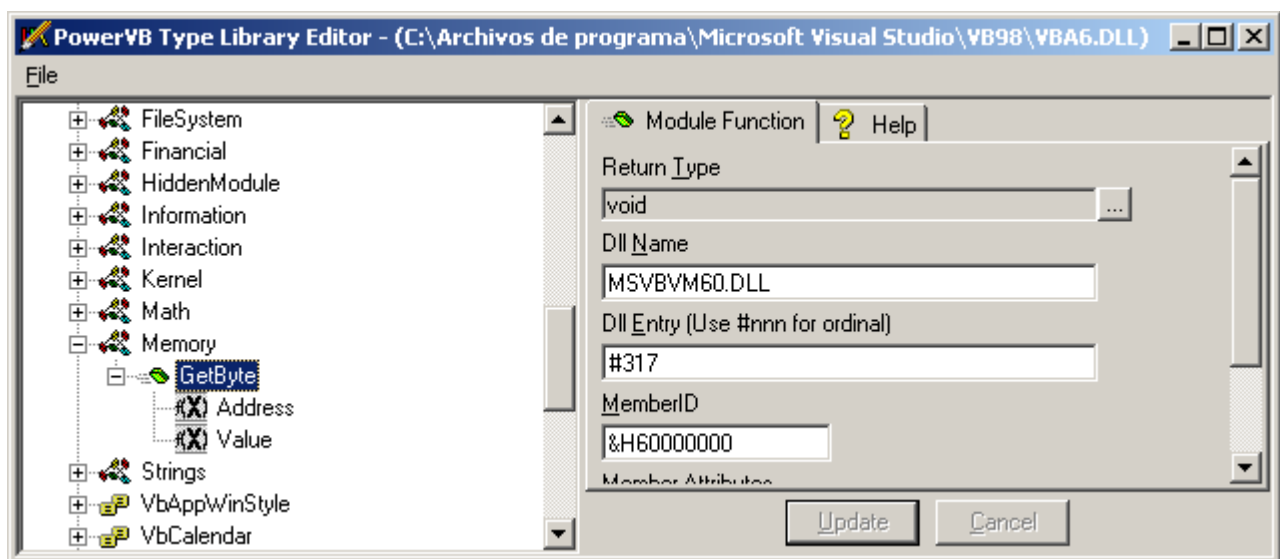
Adding new APIs.

This procedure is pretty much similar to the first one but, this time instead of adding a constant we gonna add a new API.

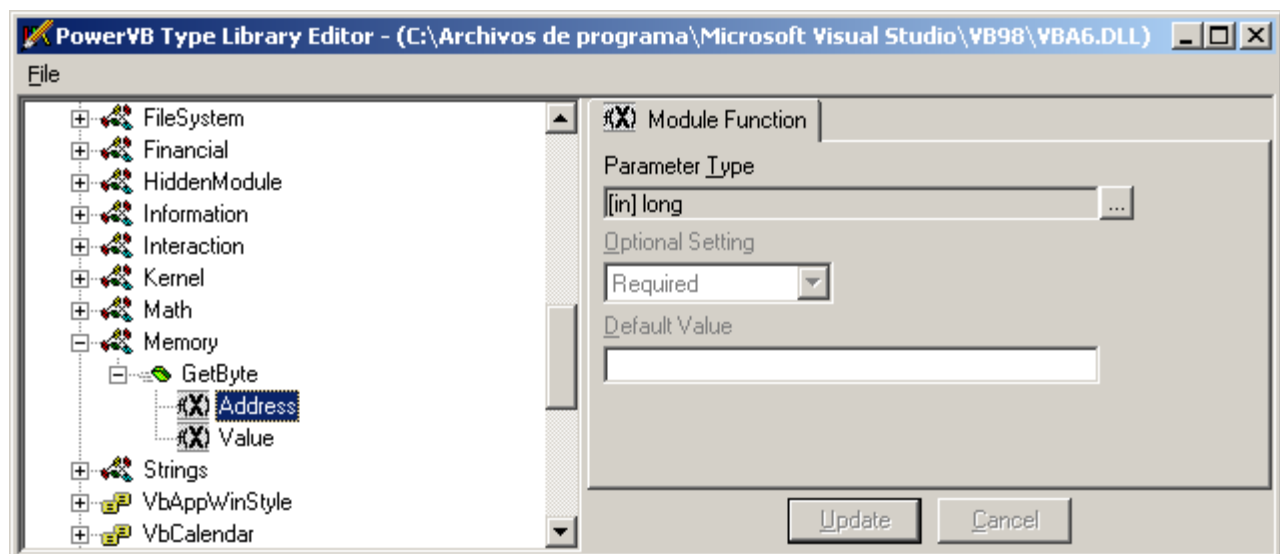
First we gonna create a new module to store all your new APIs, right click on VBA Add Type> Module , rename the module to Memory.

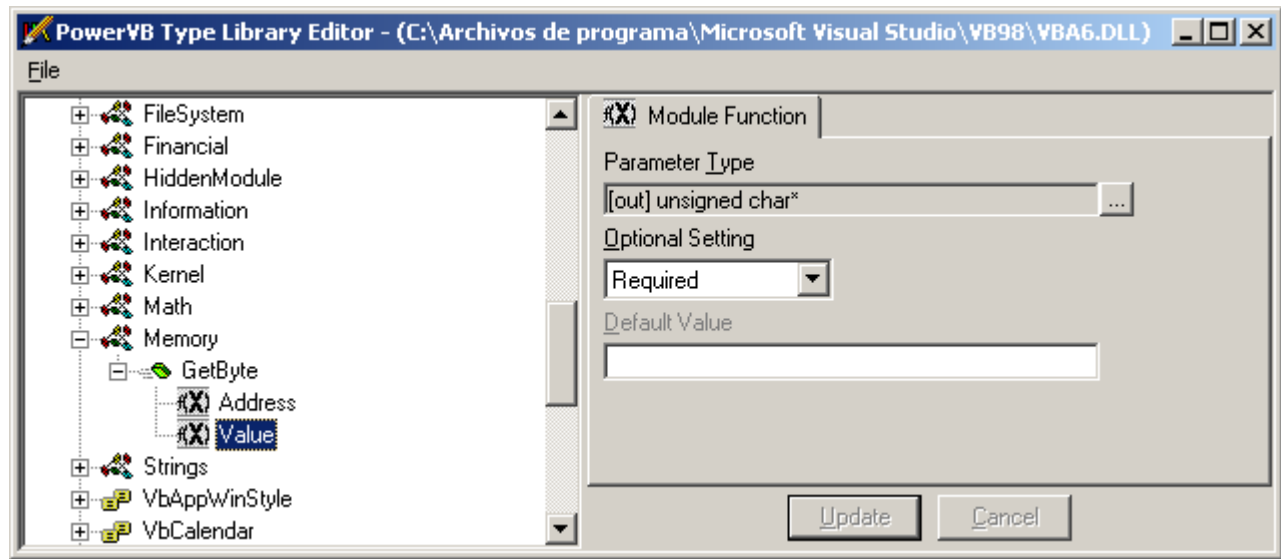
Right click on the module just created and Add Function > Function.

Now we gonna rename this new function to GetByte, this is gonna be a call to GetMem1 exported by MSVBVM60, In this picture you can see I'm using the ordinal #317 (check the text we saved at the beginning with all the exported functions to verify this)



At this point you just need to save and repeat the first process again (replace TLB), this is self explanatory, but I'm pasting some more screen caps so you can see all the values.





Using this technique you can add a whole set of memory functions, or any other api you want from MSVBVM60.

Well, I hope you find this useful or at least FUNNY !

Have funk! Cobain.