What you need to know before you start programming in VBS.

This FAQ is mostly about the pitfalls of VBS.

VBS is used in many applications. It is particularly interesting to use with WInCC Flexible because it allows you to do things that is otherwise not possible.

Do not program VBS without having accuainted yourself with the following points:

1. VBS does not support multithreading.

That means that each Script has to finish before the next can start.
And that means again, that if a Script hangs - either because of an error or because a dialog box is open - Script execution is stopped.

2. There is a script queue of 20 Scripts.

When a Script is triggered, it is added to the queue. When a script is finished it is removed from the queue.
Scripts are executed one-by-one in the order they were entered into the queue.
Additional Scripts are ignored when the queue is full.

Conclusion from 1 and 2:

It is very important to handle errors and abnormal situations correctly, otherwise script execution will stop.
One bad script can inhibit all other scripts.

The next points explains how to avoid the pitfalls.

3. Use the "err" object.

Without this, any error will stop all script execution !

There must be such a code line in the beginning of each script:

```
On Error Resume Next
```

This merely tells the script to continue with the next code line when an error is encountered.

Then after EACH code part that MAY cause problems, you must manually enter the following code lines:

```
If Err.Number <> 0 Then
  ShowSystemAlarm "Error #" & CStr(Err.Number) & " " & Err.Description
    Err.Clear
    Exit Sub
End If
```

You cannot use a message box because this is disabled in WinCC Flexible. That is why the function "ShowSystemAlarm" is used. An advantage is that the Script errors will be logged if you have setup archiving of system messages. This can help significantly in troubleshooting scripts. (Message boxes can still be made with the WScript.Shell object, but that is another story).

The "Exit Sub" is optional.
With "Exit Sub", the script is stopped, and the next script in the queue is started.
Witout "Exit Sub", the script is continued.

As to what code parts that MAY cause problems, I suggest that you use the Error handling code generously around your code until you have it debugged.

When your code runs satisfactorily under normal conditions you can remove the most of Error handling code, EXCEPT for code that reaches outside the WinCC Flexible environment.

With "reaches outside WinCC Flexible" I mean access to "objects" such as the filesystem object, the ADODB object, or the WScript.Shell object. Any other complex object should be treated as being suspect.

Allways keep in mind, that an error that isn’t handled will inhibit all script execution.

4. Use and adjust timeouts.

Many objects, but not all objects, have a timeout function. This because there may be a problem or maybe only a slow execution of the call of the object. For example, with ADODB you can access a database remotely over the LAN. But there can be a problem somewhere causing the access to SOMETIMES take seconds rather than milliseconds. The real cause can be excessive network traffic, or maintenance work on the database, or something else.
With ADODB the normal timeout is 15 seconds.
If for example, there is a cyclic script execution of logging of values every 5 seconds, and there is database problem, then it will in 100 seconds (5 seconds x 20 scripts in the queue) result in an overrun of the script queue. This will cause ALL scripts to stop functioning.
The solution is to reduce the time to less than the cyclic time of 5 seconds (3 seconds for example).

For ADODB this is done with

```
Set conn = CreateObject("ADODB.Connection")
  conn.ConnectionTimeout = 3
  conn.CommandTimeout = 3
```

For a popup box made with the WScript.Shell object, there is by default NO timeout !
So an open message box will inhibit script execution forever if the timeout isn't set.
To set a meaningful timeout you must specify for a Yes/No popup:

```
Set shellobject = CreateObject("WScript.Shell")
iTimeout = 10
shellobject.Popup strBodyText, iTimeout ,strHeader, 4+4096
```

Do not leave the timeout empty like this:

```
shellobject.Popup strBodyText, ,strHeader, 4+4096
```

"iTimeout" is the timeout in seconds. For the operator to have time to read and react the time should not be less than 10 seconds.

5. Do not use cyclically executed scripts and also scripts with the WScript.Shell popup object in the SAME project.

It can be concluded from the way VBS works, that it is possible to configure scripts that execute cyclically and scripts that display a popup window in the same project. However, it is a very bad idea to mix the two in the same project.

Pop-up windows made with WScript.Shell is special in the way that they are GUARANTEED to last for seconds because the operator has to read the text in the pop-up box and respond to it in an "intelligent" way.
Other scripts may also execute in seconds in abnormal situations, but normally they execute in milliseconds.

The effect is that as long as a pop-up window is open, cyclically triggered scripts will not be executed immediately. They will either execute later (but with other tag values than what was intended), or the queue may overrun.

6. Do not attempt to do too much with VBS.

Do not attempt to execute scripts cyclically with a cycle in the milliseconds area.

VBS in WInCC Flexible is NOT suited to be a framework to make the major part of a project.

Use the native functions in WinCC Flexible wherever possible.

The above goes for WinCC Flexible PC RT, and even more so For panels with Windows CE.

Written by Jesper M. Pedersen, july 2008.