

Single Switch Performance Test (SSPT)

System Documentation

by

Dan Yocom & Josh Huber

The Following API's are used in SSPT but are not part of the Standard Visual Basic 6 IDE... Therefore they are documented below.

Mouse Interfacing For SSPT v1.0

Win32 Hooks are a method by which you can tap into the Windows message stream for every single message directed to every window in your application. You can modify or even discard messages before they even reach the target window. This technique gives you a lot of control!

Windows Hooks are defined in the Win32 SDK as follows:

- "A Windows hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure."
- "Hooks tend to slow down the system because they increase the amount of processing the system must perform for each message. You should install a hook only when necessary, and remove it as soon as possible."

About Hooks

This section provides a detailed look into what hooks consist of, why you use them and the various types available:

1. Hooks Compared to Sub-Classing
2. An Example
3. Hook Scope: System Wide or Local Thread Hooks
4. Types Of Windows Hooks
5. Hook Chains

Hooks Compared to Sub-Classing

Essentially a Windows Hook is like a subclass except it applies at the next point up in the message chain. In Win32, messages are queued and sent to the appropriate window via the WindowProc function. A subclass can intercept messages by replacing the existing WindowProc function with a new one.

With a Hook, you get to see all messages before Windows has decided which WindowProc procedure to direct the message to. You can modify or discard most messages at this point too. So using the Hook technique you not only get to act on messages at the highest point; you can also get at messages regardless of which window the message is intended for. This is very useful if you want to process Keyboard or Mouse messages on a global basis in your application, because these messages are normally directed to the window with the focus. If you tried to do the same thing using Subclassing then you would have to subclass every single window handle in your application to follow the mouse events. Using a Hook, the process is simple!

Types Of Windows Hooks

There are 14 different types of Windows Hooks defined in Win32, however, some of these are not implemented and some cannot be used in Visual Basic or are of very limited value when not used System Wide. For this program, only the mouse hooks (WH_Mouse) are used.

The **WH_MOUSE** Hook Intercepts all mouse messages and allows you to modify or discard any message.

The vbAccelerator Windows Hooks Library

The vbAccelerator Windows Hooks Library is designed to simplify the process of working with hooks in your application. It provides a series of functions for installing and removing hooks and ensures multiple clients use the same hook function rather than installing multiple chained hooks. It is designed to be used either as a DLL or compiled directly into your application. An example of compiling the library directly into code is provided in the [vbAccelerator Accelerator control](#), so this section will cover using the DLL version.

The Hooks Library consists of two parts:

1. **IWindowsHook**

You must implement this interface to use the Hook library. Once your object implements this interface, then whenever the Hook(s) you connect to fire, the IWindowsHook_HookProc method will be called with the Hook information.

2. **GHook**

This interface allows you to attach and detach a Hook, as well as providing some helper functions for translating the IParam member of the hook function, which contains the details about the message being intercepted by the Hook.

Timing the Switch Activations

When doing time-based calculations, you have several options for tracking the amount of time that has passed between two events:

- Timer Control
- Timer object
- GetTickCount (API)
- timeGetTime (API)
- QueryPerformanceCounter (API)

The first four are not very precise, and the fourth has a tendency to suddenly jump ahead a few seconds to correct itself. So what's a programmer to do? Cheat.

You can set the timer control to an interval of one, but its accuracy is at best 55 ms. The timer function only returns hundredths. GetTickCount reports ms, but is also only accurate to 12-58 (depending on OS) ms, and timeGetTime's resolution seems to vary by machine (15ms resolution on my development rig, about 30 ms on another). QueryPerformanceCounter is incredibly precise...but can suddenly jump ahead several seconds to correct itself, and without warning.

On the surface, you might think GetTickCount or timeGetTime might be the way to go. After all, 15.625 ms should be plenty fast, right? Even 30 ms seems good. That is, until you do a little math:

1 sec / 170 ms = 5.8 Activations per Second
1 sec / 200ms = 5 Activations per Second

For example you might actually activate the switch MORE than 5.8 Activations per Second, but any results will only be updated 15-30 ms intervals, which is actually noticeable. The discrepancy will cause erroneous results. What we really need is a reliable timer with 1 ms precision or so. Fortunately, that's what we have with timeGetTime. TimeGetTime gets the elapsed time since windows started measured in milliseconds.

What's that? I said that timeGetTime is only accurate to 15 ms or so? Well, that's true...by default. It's actually accurate to 1 ms, but is updated much less often! Happily, we do have a way to correct this.

```
Public Declare Function timeBeginPeriod Lib  
"winmm.dll" (ByVal uPeriod As Long) As Long  
Public Declare Function timeEndPeriod Lib "winmm.dll"  
(ByVal uPeriod As Long) As Long
```

These two API calls were surprisingly difficult to find out about, and they're not in the [API-Guide](#).

At the beginning of your code, you make this call:

```
timeBeginPeriod 1
```

This sets the update period of timeGetTime to 1 ms.

When your program is exiting, you make this call:

```
timeEndPeriod 1
```

This restores the update period of timeGetTime to whatever it was when you started.

Between these two calls, your calls to `timeGetTime` are actually accurate to 1 ms, and all is well. For most applications, this is more than enough

Using Hotkeys to Restart Tests

Hot keys are implemented using the MCL HotKey ActiveX Control (`MCLHotkey.ocx`). `MCLHotkey.ocx` creates A system wide hotkey that is triggered when the key combination is set wether your application has the input focus or not

To use:

Add the `MCLHotkey.ocx` to your Project,

Drop the `VBHotkey` control onto your form

In the `VBHotkey1_HotkeyPressed` event add whatever code you wish to be triggered by the event

Set the control's parameters at design time

Now whenever your app is running (whether it is the foreground app or not) and the hotkey combination is triggered, your app. will execute the code in the event procedure. When your application is terminated the hotkey will be unloaded.

Parameters:

<code>AltKey</code> As Boolean	Whether the hotkey combination includes the ALT key
<code>ShiftKey</code> As Boolean	Whether the hotkey combination includes the SHIFTkey
<code>CtrlKey</code> As Boolean	Whether the hotkey combination includes the CONTROL key
<code>WinKey</code> As Boolean	Whether the hotkey combination includes the WIN key
<code>VKey</code> As KeyCodeConstants	The other key in the combination e.g <code>vbKeyF12</code> for F12 etc.

Idea's For Future Releases of SSPT

- Practice Mode
- Performance graph feature, providing for multiple test analysis.
- Enhanced note taking features (timestamping, Multiline)
- Enhanced Testing Parameter Controls
 - I. Ability to change stimulus, background, and text colors
 - II. Ability to change stimulus sounds
 - III. Ability to change number of test signals

- “Game Mode” – A fun interactive game using switch activations, releases, and holds to allow the test subject to enhance their switch skills without the monotony of the actual tests. Example: For the Activation skill have a baseball player swing a bat at a baseball.