Visual Studio Magazine Online
Classic VB Corner

Monitoring System Power Status

With the use of portable devices on the rise, you may find a need to monitor how much life is in the system your application finds itself running on.

January 5, 2010 · by Karl E. Peterson

Portable devices are becoming increasingly common. And through great strides have been made in prolonging battery life, no portable system can stay up forever as we so commonly assume when producing apps for the desktop. If you have a need to be on top of this situation, Windows is willing to provide a steady stream of notifications on power status. You only have to listen.

Windows has always been willing to tell you about itself when you ask, and this case is certainly no different. You can use the GetSystemPowerStatus API function to determine most of the relevant power settings at a glance. This function returns a SYSTEM_POWER_STATUS structure that tells you whether you're running on electricity (AC) or battery (DC).

You'll also discover what the general battery state is -- high charge, low charge, critically low charge or charging. And if the system is capable of calculating it, you'll get an estimate of the maximum possible lifetime for this battery, as well as how many minutes of remaining uptime the battery has under the current power scheme. These last numbers are definitely estimates, as anyone who's waved a cursor over the battery icon in the tray can attest.

But asking for these statistics isn't something you should have to make allowance for in a dynamic system. Recently, I showed you how to subclass ThunderMain, the hidden top-level window that Classic VB provides to every application. In addition to the broad range of general purpose notifications Windows supplies top-level windows, there is also an array of power related notifications that stream through any system that runs on DC.

After hooking into the ThunderMain message stream, your windowproc is capable of raising events for a number of useful system power alerts by watching for WM_POWERBROADCAST messages, and branching based on the wParam argument:

```
' System notification events.
Public Event PowerBatteryLow()
Public Event PowerResume()
Public Event PowerResumeAutomatic()
Public Event PowerResumeCritical()
Public Event PowerStatusChange()
Public Event PowerSuspend()
Public Event PowerSuspendQuery(Cancel As Boolean)
Public Event PowerSuspendQueryFailed()
```

```
Private Function IHookXP_Message(ByVal hWnd As Long, _
   ByVal uiMsg As Long, ByVal wParam As Long, _
   ByVal lParam As Long, ByVal dwRefData As Long) As Long

   Dim Cancel As Boolean
   Dim EatIt As Boolean
   Dim nRet As Long
   Dim msg As String

   ' Special processing for messages we care about.
   Select Case uiMsg
      Case WM_POWERBROADCAST
         ' An application should return TRUE if it processes this
         ' message.
         EatIt = True
         nRet = 1&  ' Default return value.

         Select Case wParam
            Case PBT_APMBATTERYLOW
               ' Notify applications that battery power is low.
               RaiseEvent PowerBatteryLow

            Case PBT_APMRESUMESUSPEND
               ' Notifies applications that the system has resumed
               ' operation after being suspended.
               RaiseEvent PowerResume

            Case PBT_APMRESUMEAUTOMATIC
               ' Notifies applications that the computer has woken
               ' up automatically to handle an event. Applications
               ' will not generally respond unless they handle the
               ' event, because the user is not present.
               RaiseEvent PowerResumeAutomatic

            Case PBT_APMRESUMECRITICAL
               ' Notifies applications that the system has resumed
               ' operation. This event can indicate that some or
               ' all applications did not receive a PBT_APMSUSPEND
               ' event. For example, this event can be broadcast
               ' after a critical suspension caused by a failing
               ' battery.
               RaiseEvent PowerResumeCritical

            Case PBT_APMPOWERSTATUSCHANGE
               ' Notifies applications of a change in the power
               ' status of the computer, such as a switch from
               ' battery power to A/C. The system also broadcasts
               ' this event when remaining battery power slips
               ' below the threshold specified by the user or if
               ' battery power changes by a specified percentage.
               RaiseEvent PowerStatusChange

            Case PBT_APMSUSPEND
               ' Notifies applications that the computer is about
               ' to enter a suspended state. This event typically
               ' is broadcast when all applications and installed
               ' drivers have returned TRUE to a previous
               ' PBT_APMQUERYSUSPEND event.
               RaiseEvent PowerSuspend
            Case PBT_APMQUERYSUSPEND
               ' Requests permission to suspend the computer. An
               ' application that grants permission should carry
               ' out preparations for the suspension before
               ' returning.
               RaiseEvent PowerSuspendQuery(Cancel)
               If Cancel Then
                  IHookXP_Message = BROADCAST_QUERY_DENY
               End If

         End Select
   End Select

   ' Pass back to default message handler.
   If EatIt Then
      IHookXP_Message = nRet
   Else
      IHookXP_Message = HookDefault(hWnd, uiMsg, wParam, lParam)
   End If
End Function
```

I coded this functionality into a class that I envisioned using globally within an application. Each object that needed to be alerted to power status could then simply listen in to the ongoing stream of events. This class also provides a series of methods that expose the values offered by the GetSystemPowerStatus function. For example:

```
Public Enum PowerACStatus
   ACOffline = 0
   ACOnline = 1
   ACUnknown = 255
End Enum

Public Function ACLineStatus() As PowerACStatus
   Dim sps As SYSTEM_POWER_STATUS
   If GetSystemPowerStatus(sps) Then
      ACLineStatus = sps.ACLineStatus
   End If
End Function

Public Function BatteryLifePercent() As Long
   Dim sps As SYSTEM_POWER_STATUS
   ' The percentage of full battery charge remaining. This member
   ' can be a value in the range 0 to 100, or 255 if status is
   ' unknown.
   If GetSystemPowerStatus(sps) Then
      BatteryLifePercent = sps.BatteryLifePercent
   End If
End Function

Public Function BatteryLifeTime() As Long
   Dim sps As SYSTEM_POWER_STATUS
   ' The number of seconds of battery life remaining, or –1 if
   ' remaining seconds are unknown.
   If GetSystemPowerStatus(sps) Then
      BatteryLifeTime = sps.BatteryLifeTime
   End If
End Function
```

Vista (and higher) systems offer even more detailed and responsive notifications, through the RegisterPowerSettingNotification API, which we'll dig into in the future. As always the complete code for the project described above is available on my Web site. Grab the SysInfo sample and play along as we look at all the system settings Windows willingly provides to all who listen.

About the Author

Karl E. Peterson wrote Q&A, Programming Techniques, and various other columns for VBPJ and VSM from 1995 onward, until Classic VB columns were dropped entirely in favor of other languages. Similarly, Karl was a Microsoft BASIC MVP from 1994 through 2005, until such community contributions were no longer deemed valuable. He is the author of VisualStudioMagazine.com's new Classic VB Corner column. You can contact him through his Web site if you'd like to suggest future topics for this column.