White Paper

September 5, 1997

USB - APM Interactions

USB - APM Interactions

I. APM and USB

Personal Computers today primarily follow the Advanced Power Management (APM 1.2) specification to perform platform power management. APM implementations have traditionally relied on hardware monitoring to determine how to manage the system. The Universal Serial Bus (USB) technology is now becoming an integral part of the PC platform. USB is one of the first I/O ports where several types of devices can be connected simultaneously. These devices range from keyboards to printers to video cameras. USB also relies heavily on system software to transfer the data correctly between the device and application programs.

This paper discusses the interactions between APM and USB, why the current implementations may not function efficiently together, and what can be done in the system to enable them to work together. The operating system covered in this paper is OSR2.1, the version of Microsoft Windows 95 that supports USB. It is expected that the new version of Microsoft Windows (Windows 98) will fix all of the issues described below.

The interactions described below (sections IV through VI) may or may not be issues for specific OEMs. OEMs have to decide whether or not these interactions are a problem, and then decide which solution best suits their needs.

II. APM (Advanced Power Management) Background

APM 1.2 provides an industry specification for implementing power management in the PC environment. The specification documents the BIOS calls and software interface for handling power management events and putting APM enabled hardware into a power managed state after an interval of system inactivity. The specification does not document exactly what constitutes “system inactivity”. This is left to the OEM to decide.

Systems typically measure “activity” by monitoring interrupt assertions or accesses to specific IO locations. The theory is that if devices are asserting their interrupt, or device drivers are accessing their devices then the system is “active”. Most notebooks are designed to automatically suspend when no system activity is detected. A typical suspend sequence is described in the steps below. Note that this is a high-level description of the suspend sequence and doesn’t cover every detail.

1. Once the system has determined it is inactive, it notifies the O/S through an APM request This notification is provided through a well-known APM BIOS interface, that the OS polls every second.
2. The OS power driver (VPOWERD) polls this APM BIOS function periodically (every second) looking for suspend requests.
3. When the OS detects an APM suspend request, the OS sends messages to all drivers and applications asking them if it is OK to suspend. If a driver or application rejects the suspend request, then the OS clears the APM suspend request, notifies drivers and applications that the suspend has failed, and then proceeds normally.
4. If nothing rejects the suspend request, then the OS sends another message to all drivers and applications telling them a suspend is going to happen. This gives the drivers and applications an opportunity to save any state that they may require when resuming.
5. The OS suspends the machine by calling another APM BIOS function. The APM BIOS then does the machine-specific sequence for suspending the machine.

While the machine is suspended, it monitors ‘activity’ to determine when to resume. The ‘activity’ that is monitored varies from machine to machine and can also vary depending on how ‘deeply’ the machine is suspended. For instance, a machine that has done a powered-off suspend (suspend with power removed from the chipset), may only resume when a specific button is pushed (or the lid is opened in the case of a notebook). A machine that has done a powered-on suspend (suspend with power left on the chipset), may be able to resume on interrupt activity, such as modem, keyboard or Cardbus. A typical resume sequence is described below.

1. Machine HW detects that a resume should occur. This could be a button press, a lid opening, or an interrupt asserted.
2. The APM BIOS puts all APM controlled HW back into the same state that it was in before the suspend. For a

powered-off suspend, this may mean re-initializing the device. The pre-suspend memory image is restored.

3. The OS starts running again, does OS specific initialization, and sends resume messages to all drivers and applications.
4. Drivers and applications can restore any state they need to continue operation.

The descriptions above indicate that device drivers and applications can take quite an active role in suspend and resume sequencing. In reality, most drivers and applications just ignore the power management messages.

II. USB – APM Interactions

A problem with the traditional hardware monitoring model arises because of the diversity of devices that may be connected to USB. Some of these devices (such as keyboards and mice) need to be monitored for activity with respect to system activity, and some (such as video cameras) may or may not. It depends on how a particular device is being used at a particular time.

For example, if a system has a USB keyboard attached, it should not automatically go into a power managed state if the user is typing, therefore USB activity should be monitored for system activity.

If a system has a USB video camera attached, it may be desirable to allow the system to go to sleep even though the video camera is transmitting a continuous stream of data.

If a system has both a USB keyboard and USB camera attached, the issue gets more complicated. In this case, if USB activity is monitored, the system will never go to sleep because the camera constantly transfers data. If USB activity is not monitored, the system can go to sleep while the user is typing on the keyboard. The USB system hardware cannot differentiate at the hardware level what USB device generated the data, all it can see is whether USB activity occurred or not.

III. Suspend Issues

There are two suspend issues associated with USB, depending on how system activity detection is configured on the machine. If the machine is configured to ignore USB interrupts and USB IO accesses as system activity, then the problem is **Unwanted Automatic Suspends.** This means that a USB device could be in use and yet the system believes that the machine is inactive and will request a suspend. If no drivers or applications reject this suspend request, then the machine will suspend.

If the machine is configured to recognize USB interrupts or USB I/O accesses as system activity, then the problem could be that the **Machine Never Suspends**. This is because the shipping Windows 95 OSR2.1 USB host controller driver configures the USB host controller to generate interrupts and I/O accesses to the USB controller even when there is no USB traffic, whether a USB device is connected or not. Each one of these interrupts or I/O accesses is system activity so the system always thinks that there is 'activity' in the machine.

Depending on the specific platform and its intended usage, these issues may or may not be of concern. For instance, a server class machine that never suspends for other reasons (either APM is disabled, or there are other non-USB APM activities always occurring) isn't affected. For notebooks, especially when running on battery, never auto suspending is obviously a very large concern. A typical user model for notebooks does not have a USB keyboard or mouse in use when the machine is being powered by batteries. Since the integrated keyboard and mouse are not USB devices, system activity will be detected and unwanted suspends will not occur with this user model. For desktop machines, never suspending is probably a concern given the general 'power aware' status of most consumers (both corporate and home). Unwanted suspends on desktops may or may not be a concern depending on the likelihood of a USB device being used while no other APM system activity generating peripherals are used.[1]

There are several ways that suspend issues can be dealt with. The list below gives descriptions of ways to deal with the issues, including some assessment of what impact and effects each will have.

- **Live with a risk of unwanted suspends.** Configure the platform to ignore USB interrupts and IO accesses and rely on well-behaved USB device drivers. We believe the scope of the unwanted suspend problem is limited when device drivers are written correctly. Such a driver will reject suspend requests when the device is actively being used, and can accept suspend requests when the device is inactive. A disadvantage of this model is that battery low suspend requests will also be rejected. Section V of this paper provides some guidelines for device driver writers on how to manage suspend requests from the OS. Also realize that since USB interrupts are ignored, USB devices cannot wake the system up from a suspend.

[1] Most desktops ship with standard PS2 keyboards and mice, both of which generate APM events when used.

There are still potential issues with USB keyboards and mice. In order to support user initiated suspends using the keyboard or mouse, device drivers for USB keyboards and mice would have to be designed to always accept suspend requests. This can cause a risk of an unwanted auto suspends when exclusively using a USB keyboard and/or mouse. The alternative case of having the keyboard/mouse driver reject suspend requests if the driver detects recent keyboard or mouse activity would cause the situation where a user could not use the keyboard or mouse to initiate a suspend request.

- **Live with never suspending because of inactivity.** Configure the platform to use USB interrupts or IO accesses as system activity and realize that the machine will never suspend due to inactivity. The user can still suspend the machine using a Suspend button or the Start->Suspend function on the Win95 desktop. If the suspend is a powered-on suspend, a wake event from a USB device will cause the machine to resume.
- **Acquire a modified platform specific USB host controller driver**. This driver should only generate interrupts and access specific IO locations when there is actual USB traffic. This option requires the machine to be configured to recognize USB interrupts or IO accesses as system activity in order to determine USB activity.

  Both the PIIX3 and PIIX4 are PCI devices and could be configured to share an interrupt with another PCI device. If the system is configured this way (shared interrupts) then interrupts generated by the other devices will also be considered as system activity.

  The PIIX4 has a fixed set of interrupts that it can monitor for system activity. If interrupt detection is turned on, any activity on any of these interrupts is considered APM system activity, and if interrupt detection is turned off, none of those interrupts are considered for system activity. If interrupts detection is used on PIIX4 systems there could be configurations where the CDROM driver or an IR driver consistently cause interrupts, resulting in a system that never suspends due to inactivity. (The CDROM driver polls the drive periodically if the auto-run feature is enabled, and the IR driver polls IR devices looking for input.)

  The PIIX4 can also be configured to detect specific IO accesses and use those as system activity. This feature can be used to monitor USB activity by configuring the PIIX4 to monitor IO accesses to the USB host controller's Status register, which is at offset 0x02 in the host controller's assigned IO space.
- **Disable APM.** For systems where APM functionality is not needed, APM can be disabled. The obvious impact here is that the machine will never enter any low power states. Methods for disabling APM vary from platform to platform but can typically be done by BIOS Setup options.
- **Disable USB.** For systems where APM operation is paramount, and none of the previous solutions is acceptable, then the USB host controller in the system can be disabled. This has significant end-user impact since the system has USB connectors and yet they are not functional. This solution to the issue is discouraged.

II? Resume failure after suspend (if USB controller is powered off)

When a system suspends, and the USB controller is powered off as a result, the USB devices may not be correctly restored by the Microsoft USB host controller driver after resuming. This makes USB devices unusable after the resume until they have been unplugged and replugged. The reason that the devices are not correctly restored is that the host controller driver is not aware that the host controller and the USB devices lost power. The system BIOS can leave certain registers in the host controller un-initialized, and the host controller driver will detect this and know that the host controller lost power. The host controller driver will then properly initialize the USB devices.

The specific registers that the BIOS should not restore during the resume process are:

- The Interrupt Mask register at offset 0x04 in the host controller's IO space, and
- Both Port Status and Control registers at offsets 0x10 and 0x12 in the host controller's IO space.

II? Stop Clock problem

Stop clock and stop grant are low power CPU power modes. Stop grant is less aggressive, resumes instantly, and does not affect other system activity. The CPU dissipates approximately one watt (1.0 watts) when in stop grant mode and resumes within a few clock cycles. Stop clock is more aggressive, dissipating forty milli-watts (0.040 watts) and requires approximately 1 millisecond to resume. When in stop clock mode, the CPU's clock is off, and no snooping can occur. Thus, no bus master activity can occur during stop clock.

The standard operation of the USB host controller is to access memory as a bus master every millisecond looking for

work to do. This memory polling disallows effective usage of Stop Clock mode for power management.

For systems where the aggressive power savings of Stop Clock mode are required, a modified USB host controller driver must be used. This driver should turn off host controller polling when there are no USB devices attached or when a device is attached but not transmitting data.. When devices are connected and transmitting data, stop grant will need to be used.

III? USB Device Drivers

Device drivers for USB devices are situated for determining whether or not there is USB activity. The Windows Driver Model (WDM) allows device drivers to either accept or reject Power Management suspend requests. Device drivers can make the decision on whether to accept or reject a suspend request based on the device's current or recent activity.

Guidelines for whether or not a suspend request should be rejected will vary depending on the device function. In general, suspend requests should be rejected if the device is actively being used and should be accepted if the device is idle. For instance, a USB printer driver would reject suspend requests if the printer was being used, and accept the request if the printer is idle.

A USB video camera provides an interesting case where it's not clear when the driver should reject a suspend request. The user could leave his machine with a window showing the local video. This window would be constantly displaying data from the camera, but the user may not be at the machine. In this case a suspend request could reasonably be accepted even thought the camera was actively transferring data.

If the user is doing video conferencing there would be other subsystems (audio, modem) active that indicate that there is active use of the machine and may count as APM system activity or have drivers that will reject a suspend request. However, if none of those subsystems will cause the suspend to be rejected, the user could be faced with a suspending machine even though actively involved in a video conference. This would suggest that the USB camera driver should reject suspends anytime the camera is active.

It should be noted that device drivers cannot distinguish between suspend requests coming from the user (by hitting a platform specific 'Suspend' button or by doing the normal Start->Suspend sequence), those generated by the system as a result of 'system inactivity', and battery low suspend requests. If the user requests a suspend and it is rejected by a device driver, a message will appear indicating that some application or device has rejected the suspend, and the user should stop that application or device before the suspend can happen.

Drivers for USB keyboards and mice should always accept suspend requests. This is because a mouse or a keyboard can be used when the user wants the machine to suspend, and if these drivers rejected the suspend because the mouse or keyboard was recently active, the user could never suspend the machine. Note that this means on systems where a USB keyboard and/or mouse is being used, and there is no other system activity the machine could suspend even though the keyboard and/or mouse are actively being used.

When a device driver accepts a suspend request, it should prepare for the device to be suspended which may mean saving some device state. On some systems (both desktop and mobile), the device may actually lose power when the system suspends. If this happens, when the system resumes the driver will be notified that the device has been unplugged and then replugged. The handling of this should be the same as if the user had unplugged and replugged the device while the system was active.

Vendors writing OSR2.1 device drivers for USB devices should validate that their drivers will continue to work with Windows98. The power management policies provided by Windows98 are different from those in OSR2.1 and may require slight driver modification to get the best functionality. Information on Windows98 power management can be found on the Microsoft web pages at http://www.microsoft.com.