

PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	Unoccupied Slot Power Hand-off State Clarification
DATE:	January 23, 2007
AFFECTED DOCUMENT:	PCI Firmware Specification, Revision 3.0
SPONSOR:	Dong Wei; Hewlett Packard Company

<u>Part I</u>

1. Summary of the Functional Changes

This ECN clarifies the unoccupied slot power state at hand-off.

2. Benefits as a Result of the Changes

This ECN allows the unoccupied slots' power to be off at hand-off, which is a reasonable implementation and many systems implement that way today.

When the PCI Hot Plug spec was originally designed, it wanted the ROM to leave unoccupied slots (with closed MRL) on, in case the slot was really occupied by a card that didn't show its presence (a technical violation of the spec, but a violation that was not enforced on non-hot-plug slots), and then the user loaded an OS that did not have hot-plug support to turn it back on. This is no longer seen as an issue today, and it is better dealt with in implementation specific ways in any case.

3. Assessment of the Impact

This ECN has no impacts to the existing hardware and software. It provides a clarification to the power state at hand-off for the unoccupied slots.

4. Analysis of the Hardware Implications

No impact to underlying hardware.

5. Analysis of the Software Implications

No impact to software.

<u>Part II</u>

Detailed Description of the change

Note: Specification text is based on PCI Firmware Specification, Version 3.0, dated June 20, 2005 . All page numbers and section numbers mentioned below are based on that specification.

3.5 Device State at Firmware/Operating System Handoff

System firmware is only required to configure the boot and console devices. This section specifies the state of the PCI subsystem at firmware handoff and provides guidance to the operating system on how to determine if a particular component was configured by firmware. PCI subsystem refers to components that are compliant to the PCI, PCI-X, or PCI-Express Specifications. In this section, "PCI Specifications" refers to the PCI, PCI-X, or PCI Express Specification.

Firmware owns the PCI subsystem prior to handing control off to the operating system. The handoff point is the return from EFI ExitBootServices(). After the operating system loader calls ExitBootServices(), the operating system owns the PCI subsystem.

Firmware may provide pre-boot user interaction to allow the system operator to specify the desired boot and console devices.

Firmware shall configure the entire path to the console (both input and output) and boot devices. This includes, the chipset, bridges, and multi-function devices. The device configuration is required to load the operating system loader, display boot up messages, and allow operator interaction with the boot process.

Optionally, firmware may configure all devices and bridges in the system. Firmware is not required to configure devices other than boot and console devices.

Since not all devices may be configured prior to the operating system handoff, the operating system needs to know whether a specific BAR register has been configured by firmware. The operating system makes the determination by checking the I/O Enable, and Memory Enable bits in the device's command register, and Expansion ROM BAR enable bits. If the enable bit is set, then the corresponding resource register has been configured.

Note: The operating system does not use the state of the Bus Master Enable bit to determine the validity of the BARs. If the BAR ranges are enabled, the device must respond to those addresses. The device may not be able to master a transaction, but enabled BARs shall be configured correctly by firmware.

The operating system is required to configure PCI subsystems:

- During hotplug
- For devices that take too long to come out of reset
- □ PCI-to-PCI bridges that are at levels below what firmware is designed to configure

Firmware must configure all Host Bridges in the systems, even if they are not connected to a console or boot device. Firmware must configure Host Bridges in order to allow operating systems to use the devices below the Host Bridges. This is because the Host Bridges programming model is not defined by the PCI Specifications. "Configured" in this context means that:

- □ Memory and I/O resources are assigned and configured.
- □ Includes both the resources consumed by the Host Bridge and the resources passed through to the secondary bus.
- □ The bridge is enabled to receive and forward transactions.
- □ The bridge is operating in "safe" mode. Safe mode includes:
 - Enabling resources such as: I/O Port, Memory addresses, VGA routing, bus number, etc.
 - Enabling detection of parity and system errors
 - Programming cacheline, latency timer, and other registers as required by the PCI Specifications.

Firmware must report Host Bridges in the ACPI name space. Each Host Bridge object must contain the following objects:

- □ _HID and _CID
- _CRS to determine all resources consumed and produced (passed through to the secondary bus) by the host bridge. Firmware allocates resources (Memory Addresses, I/O Port, etc.) to Host Bridges. The _CRS descriptor informs the operating system of the resources it may use for configuring devices below the Host Bridge.
 - _TRA, _TTP, and _TRS translation offsets to inform the operating system of the mapping between the primary bus and the secondary bus.
- **_**___PRT and the interrupt descriptor to determine interrupt routing.
- □ _BBN to obtain a bus number.
- □ _UID to match with EFI device path.
- □ _SEG if it has a non-zero PCI Segment Group number.
- □ _STA if hot plug is supported.
- □ _MAT if hot plug is supported.

Firmware is required to configure all PCI-to-PCI Bridges in the hierarchy leading to boot and console devices. Firmware may optionally configure all PCI-to-PCI Bridges in the system. When configuring a PCI-to-PCI Bridge, Firmware must set it to safe mode. This includes:

- Programming and enabling resources such as: I/O Port, Memory addresses, VGA routing, bus number, etc.
- □ Enabling detection of parity and system errors
- □ If applicable, program cache_line, latency timer, and other registers as required by the PCI Specifications.

□ If applicable¹, disable Discard SERR# Enable. The Discard Timer SERR# Enable bit in the Bridge Control Register controls whether the timer waiting for the completion of a delayed transaction generates an SERR (value=1) or simply discards the transaction (value=0) on a time-out. The value of 1 is generally required when peer-to-peer transactions are allowed to and from cards under that bridge. Allowing peer-to-peer transactions is operating system policy and may not be supported on all platforms. Therefore, the bit should be set to 0 when firmware hands off to the operating system, and any changes to the setting to support peer-to-peer under the PCI-to-PCI Bridges should be made by the operating system.

The operating system may provide software to configure PCI-to-PCI bridges for optimum performance.

All slots with an open MRL must be disabled and their Power Indicators must be turned off. All occupied slots with MRL closed must be enabled and their Power Indicators must be turned on. The slot power state for unoccupied slots with MRL closed is implementation dependent and their Power Indicators must reflect the slot power state.

Firmware must deassert RST# for all occupied PCI slots below the Host Bridge. Firmware must observe required wait times such as Trhfa (RST# High to First configuration Access) after taking a bus out of reset. Firmware only needs to delay once for all PCI bus controllers before handing control to the operating system. This allows the operating system to successively walk PCI buses without having to successively delay (post reset quiesce period, 1 second) for each bus.

PCI-to-PCI Bridges have RST# asserted by default for the secondary bus. Firmware is not required to deassert RST# on secondary buses that are not used for boot and console devices.

EFI drivers and applications shall not change BAR assignments. The PCI BARs (Base Address Registers) and the configuration of any PCI-to-PCI bridge controllers belong to the firmware component that configured the PCI Bus prior to the execution of the device driver.

The operating system must not assume that all devices have been configured. Per Section 2.5.6 of EFI (rev 1.10): the presence of an EFI driver in the system firmware or in an option ROM does not guarantee that the EFI driver will be loaded, executed, or allowed to manage any devices in a platform. In addition, EFI drivers are not involved during PCI hot plug.

Note: The operating system does not have to walk all buses during boot. The kernel can automatically configure devices on request; i.e., an event can cause a scan of I/O on demand.

The operating system can determine if the device's BARs (Base Address Registers) have been configured by firmware by checking the I/O Enable, and Memory Enable bits in the device's command register, and Expansion ROM BAR enable bit. If the enable bit is set, then the corresponding resource has been configured. If the enable bit is not set, the operating system cannot assume that the associated BAR register contains valid information.

The address that a processor uses to access a device is not necessarily the same as the address stored in the device's BAR. The translation (_TRA, _TTP, and _TRS) information is not available to the operating system until after the operating system brings up the ACPI interpreter. The operating system must wait until after the ACPI interpreter is up to determine the address at the processor side associated with the BAR registers configured by firmware. Before re-enabling a resource, the operating system must reprogram the BAR register using a value that falls within the range reported in the _CRS descriptor of the parent Host Bridge. The operating system must ensure that the address range is not used by any other device below that Host Bridge.

¹ For example, does not apply to PCI Express.

Operating systems must not configure devices with resources outside what is reported by the Host Bridge _CRS. Firmware reports the address ranges that are routed to that particular Host Bridge. There is no guarantee that devices under that bridge will respond to other address ranges.

The Expansion ROM BAR (at 0x30) is normally not enabled at the firmware handoff to the operating system and the operating system must assume that the BAR content is invalid. For some devices, when the Expansion ROM BARs are enabled, the device's other BARs are disabled. PCI specifies that a device may share decoders between the Expansion ROM BAR and other BARs, and that device independent software must not access the other BARs when the Expansion ROM BAR is enabled. Firmware may leave the Expansion ROM BARs enabled if it happens to know that the device does not share address decoders. This could be firmware based on the Device ID or firmware that is shipped in the card itself. The device via the other BARs. If the operating system wants to use the Expansion ROM, it must "take turns" enabling the Expansion ROM BAR, using the ROM, then disabling the BAR again before resuming access to the card via its other BARs. In other words, the operating system shall not assume that the card has dual decoders and that the Expansion ROM BAR content is correct.