



## PCI-SIG ENGINEERING CHANGE NOTICE

<b>TITLE:</b>	Enhanced Configuration Access Mechanism Options
<b>DATE:</b>	January 7, 2004
<b>AFFECTED DOCUMENT:</b>	PCI-X Protocol, Rev 2.0a
<b>SPONSOR:</b>	Dong Wei, Hewlett-Packard Company

### **Part I**

#### **1. Summary of the Functional Changes**

Make the enhanced configuration access mechanism required for PC-compatible platforms, but optional for platforms based on other processor/system architectures where firmware abstractions are provided for the configuration space access (e.g., DIG64-compliant systems).

Make the number of bits used for bus numbers (when mapping from memory address to Configuration Space) flexible. This removes the implied 256MB alignment and makes a system only needing to allocate address space needed for the bus numbers it supports. This makes the currently defined enhanced configuration mechanism more flexible/friendly as far as implementation is concerned.

#### **2. Benefits as a Result of the Changes**

This change makes the approach of the enhanced configuration access mechanism defined in PCI-X 2.0 consistent with that of the original (CF8/CFC) configuration access mechanism defined in PCI 2.3. In PCI 2.3, the CF8/CFC mechanism is required only on PC-compatible systems. Other systems, particularly those that do not rely on shrink-wrapped operating systems or systems that require a standard firmware-based SAL (e.g., DIG64-compliant systems) are not burdened with unnecessary hardware-level standards.

This change also provides the flexibility of the alignment of the enhanced configuration address space for the small PC-compatible systems.

#### **3. Assessment of the Impact**

This is a minor addition to the existing revisions of the specifications, comparable to the conventional configuration access mechanism statement in PCI 2.3.

There is no impact to systems that currently conform to the PCI-X specification.

It is also allowed for DIG64 compliant systems to implement the mechanism as describe in Section 7.4.1, it is up to the designers to choose, but the specification does not need to require such implementation because the OS accesses the enhanced configuration space through SAL calls.

#### **4. Analysis of the Hardware Implications**

There is no impact to existing hardware.

This allows non PC-compatible systems to be able to more freely leverage the existing chipsets or interoperate with existing chipsets.

The alignment flexibility provided by this change also provides small systems better flexibility in their design to make more memory available to the customers.

## **5. Analysis of the Software Implications**

A device driver should use the API provided by the operating system to access the Configuration Space of its device and not directly by way of the hardware mechanism, so there is no impact to the device drivers.

Operating system software on PC-compatible systems can still use the direct hardware mechanism to access the Extended Configuration Space as is.

Operating system software on DIG64-compliant systems have been using firmware abstractions to access the conventional configuration space and will continue to do so for the Extended Configuration Space, so there is no need for a specified hardware access mechanism.

The alignment flexibility provided is fully covered by the ACPI interfaces defined in the PCI Firmware Specification 3.0 for the PC-compatible systems.

## Part II

### Detailed Description of the change

Make the changes (in red color) shown below (leveraged from the Conventional PCI 2.3 Spec), including the reference footnote:

#### 7.4.1 Enhanced Configuration Access Mechanism

For systems that are PC-compatible or that do not implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the enhanced configuration access mechanism is required as defined in this section.

For systems that implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the operating system uses the standard firmware interface, and the hardware access mechanism defined in this section is not required. For example, for systems that are compliant with *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64), Version 2.1 (DIG64 2.1)*,<sup>1</sup> the operating system uses the SAL firmware service to access the Configuration Space.

In all systems, device drivers are encouraged to use the application programming interface (API) provided by the operating system to access the Configuration Space of its device and not directly use the hardware mechanism.

The enhanced configuration access mechanism utilizes a flat memory-mapped address space to access device configuration registers. The memory address determines the configuration register accessed, and the memory data updates (for a write) or returns the contents of (for a read) the addressed register.

For those systems that implement the enhanced configuration access mechanism, PCI-X Mode 2 host bridges translate the memory-mapped Configuration Space accesses from the host processor to configuration transactions as defined in Table 7-4. The mapping from memory address A[27::0] to Configuration Space is defined in Table 7-4. The base address A[63::28] is allocated in an implementation-specific manner and reported by the system firmware to the operating system.

The size and base address for the range of memory addresses mapped to the Configuration Space are determined by the design of the host bridge and the firmware. They are reported by the firmware to the operating system in an implementation-specific manner. The size of the range is determined by the number of bits that the host bridge maps to the Bus Number field in the configuration address. In Table 7-4, this number of bits is represented as  $n$ , where  $1 \leq n \leq 8$ . A host bridge that maps  $n$  memory address bits to the Bus Number field supports bus numbers from 0 to  $2^n - 1$ , inclusive, and the base address of the range is aligned to a  $2^{(n+20)}$  byte memory address boundary. Any bits in the Bus Number field that are not mapped from Memory Address bits must be set to zero.

For example, if a system maps three Memory Address bits to the Bus Number field, the following are all true:

---

<sup>1</sup> *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64), Version 2.1, January 2002, www.dig64.org*

- ❑  $n = 3$ .
- ❑ Address bits A[63::23] are used for the base address, which is aligned to a  $2^{23}$  byte (8 MB) boundary.
- ❑ Address bits A[22::20] are mapped to bits [2::0] in the Bus Number field.
- ❑ Bits [7::3] in the Bus Number field are set to zero.
- ❑ The system is capable of addressing bus numbers between 0 and 7, inclusive.

A minimum of one Memory Address bit ( $n = 1$ ) must be mapped to the Bus Number field. Systems are encouraged to map additional Memory Address bits to the Bus Number field as needed to support a larger number of buses. Systems that support more than 4 GB of memory addresses are encouraged to map eight bits of Memory Address ( $n = 8$ ) to the Bus Number field. Note that in systems that include multiple host bridges with different ranges of bus numbers assigned to each host bridge, the highest bus number for the system is limited by the number of bits mapped by the host bridge to which the highest bus number is assigned. In such a system, the highest bus number assigned to a particular host bridge would be greater, in most cases, than the number of buses assigned to that host bridge. In other words, for each host bridge, the number of bits mapped to the Bus Number field,  $n$ , must be large enough that the highest bus number assigned to each particular bridge must be less than or equal to  $2^n - 1$  for that bridge.

As for any PCI-X configuration transaction, configuration transactions generated by the enhanced configuration mechanism are DWORD transactions (transferring up to a single naturally aligned DWORD) and do not support exclusive access (LOCK#).

The enhanced configuration access mechanism operates independently from the mechanism defined in PCI 2.3 for generation of configuration transactions. The logic in the host bridge that generates configuration transactions using one method must not interfere in any way with logic that generates configuration transaction using the other method.

**Table 7-4: Enhanced Configuration Access Address Mapping**

<b>Memory Address (Note 1)</b>	<b>Configuration Space (Note 2)</b>
A[ <u>27(20+n-1)::20</u> (Note 3)]	Bus Number (Note 3)
A[19::15]	Device Number
A[14::12]	Function Number
A[11::8]	Upper Register Number
A[7::2]	Lower Register Number

**Notes:**

1. Shows the address bits of the memory transaction before it is translated into a configuration transaction.
2. Shows the fields of the configuration address into which the memory address bits are mapped. See Figure 2-21 for the format of the configuration address on the PCI bus.
3. The number of bits the system maps from Memory Address to the Bus Number field is determined by the design of the host bridge, and is expressed as  $n$ , where  $1 \leq n \leq 8$ . Any bits in the Bus Number field that are not mapped from Memory Address bits must be set to zero.

The system hardware must provide a method for the system software to guarantee that a write transaction using the enhanced configuration access mechanism is completed by the completer before system software execution continues.