



PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	16-bit PCI-X
DATE:	November 20, 2003
AFFECTED DOCUMENT:	PCI-X Protocol Specification, Revision 2.0 (and 2.0a)
SPONSOR:	Stillman Gates; Adpatec

Part I

1. Summary of the Functional Changes

Change 16-bit PCI-X from required for Mode 2 devices to optional.

2. Benefits as a Result of the Changes

High-performance PCI-X devices that have no expectation of ever being used in a 16-bit application would be freed of the design and test burden of the 16-bit feature.

3. Assessment of the Impact

All designs compliant with the present version of the specification remain compliant. The feature simply becomes optional.

4. Analysis of the Hardware Implications

There is no impact on bridges, since the feature was already optional for them.

For devices, market forces will determine when the cost of designing and testing this feature are appropriate. High-performance designs are relieved of the burden if it is unlikely to be used.

5. Analysis of the Software Implications

None.

Part II

Detailed Description of the change

Change page 19 as follows:

To achieve these higher data rates, PCI-X 2.0 adds a source-synchronous clocking mode for the highest-performance burst transactions that enables transfer rates in excess of 4 GB/s. It also defines:

- ❑ 1.5V signaling levels for faster signaling and improved signal integrity characteristics
- ❑ Error correcting codes (ECC) for improved error immunity
- ❑ Expanded Configuration Space address range of 4096 bytes
- ❑ An optional low-pin-count 16-bit interface for embedded applications
- ❑ A new device-ID messaging transaction for peer-to-peer communication

Following the well-established tradition of all PCI specifications, PCI-X 2.0 stresses backward compatibility. Devices and cards compliant with PCI-X 1.0b are fully supported in PCI-X 2.0, in what is defined as Mode 1. The higher transfer rates in PCI-X 2.0 are defined as Mode 2. PCI-X devices that support both Mode 1 and Mode 2 are completely interoperable with devices designed to support the previous revision of the PCI-X definition and 33 MHz, 3.3V conventional PCI devices. They optionally support the 66 MHz conventional PCI mode. ECC support, and the expanded Configuration Space, ~~and the 16-bit interface~~ are required for Mode 2 devices. The optional 16-bit interface is defined only for Mode 2 devices. Forwarding device ID messages is required for PCI-X Mode 2 bridges. ECC support and device ID messaging are optional for Mode 1 devices.

Change page 22 as follows:

Mode 2 also includes the following features:

1. Almost all of the Mode 1 control signal and transaction phase rules have been kept the same in Mode 2 to minimize design changes.
2. The initiator of transactions that use the highest performance commands drives data at two or four times the common clock frequency. These cases use source-synchronous clocking techniques for transferring the data, rather than capturing data with the common clock.
3. The Configuration Space addresses are extended from eight to 12 bits, enabling the addressing of 4096 bytes for each function of each device.
4. An optional low-pin-count 16-bit interface for embedded applications.

Change page 23 as follows:

Table 1-2 shows the supported combinations of modes and features for PCI-X.

Table 1-2: Supported PCI-X Modes and Features

Feature	PCI-X Mode 1	PCI-X Mode 2
Device types	PCI-X 66 PCI-X 133	PCI-X 266 PCI-X 533
Bus width	64 bit optional 32-bit required	64 bit optional 32-bit: Required for add-in card applications. Optional for embedded applications 16-bit required optional
Interface Signaling	3.3V	1.5V and 3.3V
Error protection	Parity required ECC optional	ECC required
Data capture	Common-clock	Common-clock and source-synchronous
12-bit Configuration Space address	Not supported	Required
Device ID messages	Optional	Required

Change page 47 as follows:

The following PCI-X requirements are different from conventional PCI:

1. All devices that initiate memory transactions must be capable of generating 64-bit addresses.
2. Each device includes a configuration status bit that indicates the width of that device's interface.
3. In ECC mode, the upper bus is only used during 64-bit data phases. See Section 2.12.1.3 for details.
4. All Mode 2 devices ~~(other than bridges) are required to optionally~~ support a 16-bit interface. Mode 2 bridges never support a 16-bit interface on their primary bus and optionally support a 16-bit interface on their secondary bus. The 16-bit interface is intended for embedded applications, so no width negotiation protocol or add-in card connector pin-out is specified, and bridge hierarchies are not allowed. The 16-bit bus always operates in Mode 2, and all transactions are 16 bits wide.

Change page 59 as follows:

9. All Mode 2 devices ~~are required to~~ optionally support a 16-bit interface. Mode 2 bridges never support a 16-bit interface on their primary bus and optionally support a 16-bit interface on their secondary bus.
 - a. The 16-bit interface is intended for embedded applications, so no width negotiation protocol or add-in card connector pin-out is specified, and bridge hierarchies are not allowed.
 - b. The 16-bit bus always operates in Mode 2 and all transactions are 16-bits wide.
 - c. In 16-bit transfers, there are twice as many address and attribute phases as 64- and 32-bit transfers.
 - d. As in 64- and 32-bit transfers, the minimum data granularity (ignoring byte enables) is one DWORD. So 16-bit common-clock data phase and source-synchronous subphases always come in pairs that are aligned to a DWORD boundary.
 - e. 16-bit transfers exclusively use seven-bit ECC plus an additional special check bit.

Change page 173 as follows:

2.12.2 16-Bit Bus Width

PCI-X Mode 2 devices (other than bridges, which are described below) ~~are required to~~ optionally support a 16-bit version of the AD bus. A device designed exclusively for embedded applications (not to connect directly to an add-in card connector) is permitted to support only a 16-bit interface. (64- and 32-bit support is not required for devices designed exclusively for embedded applications.)