C H A P T E R 9

PCI

This chapter presents the PC 98 requirements and recommendations for Peripheral Component Interconnect (PCI) host controllers and peripherals.

The PCI architecture has become the most common method used to extend PCs for add-on adapters. Windows and Windows NT use the basic PCI infrastructure to gain information about devices attached to the PCI bus. The ability of PCI to supply such information makes it an integral part of the Plug and Play architecture in Windows.

Specific requirements related to PCI are defined in the following chapters:

- Requirements for dual IDE adapters that use PCI are defined in the "IDE and ATAPI" chapter in Part 3 of this guide.
- Requirements for PCI-to-PC Card bridges are defined in the "PC Card" chapter in Part 3 of this guide.
- Requirements for graphics devices that use PCI are defined in the "Graphics Adapters" chapter in Part 4 of this guide.
- Requirements for audio implementations that use PCI are defined in the "Audio Components" chapter in Part 4 of this guide.

Contents

PCI Basic Requirements

This section summarizes the basic design requirements for PCI.

1. All components comply with PCI 2.1
Required
All cards, bridges, and devices that use PCI must be designed to meet the requirements defined in *PCI Local Bus Specification, Revision 2.1* (PCI 2.1).

Compliance with this requirement is demonstrated based on the compliance process of the PCI Special Interest Group (SIG).

2. System does not contain ghost cards
Required
A computer must not include any ghost cards, which are cards that do not decode the type 1/type 0 indicator. Such a card will appear on bus 0 as all the buses behind it that use the same IDSEL. Notice that it is acceptable, as defined in PCI 2.1, for a single-function card to decode the IDSEL and AD[1::0] pins and not decode AD[10::8] if the card does not have bit 7 set in the header type. This requirement also excludes, for example, devices that ignore some type 0 transaction bits and therefore appear at multiple device/function addresses.

A PCI card should be visible through hardware configuration access at only one bus/device/function coordinate.

3. System uses standard method to close BAR windows on nonsubtractive decode PCI bridges
Required
PCI-to-PCI bridges must comply with the *PCI to PCI Bridge Specification, Revision 1.0.* Setting the base address register (BAR) to its maximum value and the limit register to zeros should effectively close the I/O or memory window references in that bridge BAR.

4. System supports PCI docking through a bridge connector
Recommended

It is recommended that the system support docking through a bridge connector, with the actual bridge on the docking station, not on the mobile unit. The bridge can be positive or subtractive decoding. The bridge should create a new bus number so devices behind the bridge are not on the same bus number as other devices in the system.

After a warm dock, the BIOS should not configure the bridge or any other devices in the docking station. That is the responsibility of the operating system.

The PCI-to-ISA bridge should be placed on the docking station, not on the mobile unit. Mobile PCs typically do not have ISA expansion slots, and the ISA devices on the mobile PC can be controlled by the Plug and Play interface. For more information on requirements for docking station systems, see the “Basic PC 98” chapter in Part 2 of this guide.

Notice that implementing delayed transactions for PCI-to-PCI and PCI-to-ISA docking bridges is required in PCI 2.1 only when certain timing conditions are not met. For PC 98 design requirements, PCI 2.1 is interpreted to mean that delayed transactions are required only when “targets cannot complete the initial data phase within the requirements of this specification” (as stated in PCI 2.1). Delayed transactions are a hardware-related timing issue (and will provide a performance advantage), but are not related to operating system requirements.

5. PCI chip sets support Ultra DMA/33
Required

PCI chip sets must implement DMA as defined in SFF 8020i and must implement Ultra DMA/33 (also known as Ultra-ATA) as defined in the specification submitted by Quantum Corporation for inclusion in the ATA-4 specification.

Ultra DMA/33 is required to avoid the bottleneck created by the current 16.6-Mb/s limit on disk transfer. Ultra DMA/33 also provides error checking for improved robustness over previous IDE implementations.

PCI Controller Requirements

This section summarizes PCI controller requirements.

6. System-board bus complies with PCI 2.1
Required

The system-board bus hardware should comply with PCI 2.1. The bus design must fully implement all bus requirements on every expansion card connector.

7. Bus master privileges are supported for all connectors
Required

To ensure full Plug and Play functionality on a PCI bus with expansion cards, all PCI connectors on the system board must be able to allow any PCI expansion card to have bus master privileges.

8. ISA Write Data Port address is propagated to the ISA bus at power up
Required

If the system uses an ISA bus in conjunction with a PCI bridge, the Plug and Play ISA Write Data Port address must be propagated at all times through the bridge to all ISA buses that might contain external ISA Plug and Play cards. The address must be propagated at power up and system reset. This ensures that the system can identify, isolate, and configure external Plug and Play ISA cards plugged into the ISA bus during the boot process.

9. Functions in a multifunction PCI device do not share writable PCI Configuration Space bits
Required

The operating system treats each function of a multifunction PCI device as an independent device. As such, there can be no sharing between functions of writable PCI Configuration Space bits (such as the Command register).

Notice that the PC Card 16-bit Interface Legacy Mode BAR—offset 44h in the Type 2 PCI header—is the only exception to this requirement. This register must be shared between the two functions, just as they must share the same compatibility registers with the Exchangeable Card Architecture (ExCA) programming model, as defined in the *PCI to PCMCIA CardBus Bridge Register Description* (Yenta specification), by Intel.

For more information about design requirements for CardBus controllers, see the “PC Card” chapter in Part 3 of this guide.

Plug and Play for PCI Controllers and Peripherals

This section summarizes the Plug and Play requirements for PCI devices.

10. Devices use PCI 2.1 Configuration Space register for Plug and Play device ID

Required

The PCI 2.1 specification describes the Configuration Space register used by the system to identify and configure each device attached to the bus. The Configuration Space register is made up of a 256-byte field for each device and contains sufficient information for the system to identify the capabilities of the device. Configuration of the device is also controlled from this register.

The Configuration Space register is made up of a header region and a device-dependent region. Each Configuration Space register must have a 64-byte header at offset 0. All the device registers that the device circuit uses for initialization, configuration, and catastrophic error handling must fit in the space between byte 64 and byte 255.

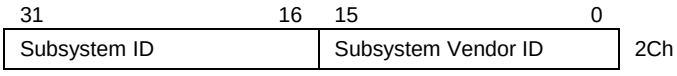
All other registers that the device uses during normal operation must be located in normal I/O or memory space. Unimplemented registers or reads to reserved registers must complete normally and return zero (0). Writes to reserved registers must complete normally, and the data must be discarded.

11. Device IDs include PCI 2.1 Subsystem IDs

Required

The following diagram shows the two registers added to the Configuration Space header for PCI 2.1. Although these registers are only recommended in PCI 2.1, they are mandatory for PC 98. Support for these registers requires non-zero values to be populated for both the Subsystem ID and Subsystem Vendor ID.

New registers in Configuration Space header for PCI 2.1



These fields are necessary for the correct enumeration of a device. When the Subsystem ID fields are populated correctly for the adapter, Windows can differentiate between adapters based on the same PCI chip.

The Subsystem ID also allows Windows to load system miniports for system-board devices. Thus, Subsystem IDs are also a requirement on system-board devices. The exceptions to this requirement are PCI-to-PCI bridges and core chip sets.

Two methods can be used to implement a Subsystem Vendor ID:

- Load the value by hardware methods—for example, pin strappings at RST, an attached parallel or serial ROM, and so on.
- Program the Subsystem Vendor ID using BIOS. Two designs using the BIOS method meet PC 98 requirements:
  - Make a copy of the Subsystem Vendor ID in PCI user-defined space. Any writes to this location will change both the copy and the Subsystem Vendor ID field. Any writes to the Subsystem Vendor ID are discarded.
  - Make a write-enable bit in the PCI user-defined space. The BIOS can turn this bit on, change the Subsystem Vendor ID, and then turn it off.

For more information, see the article titled “IDs and Serial Numbers for Plug and Play” on the web site at http://www.microsoft.com/hwdev/busbios/.

Important: Multiple-monitor support allows display class devices to be initialized independently of the system initialization process. For this reason, system-board and add-on display devices cannot use the VGA BIOS POST routine to populate the Subsystem Vendor ID because the device’s POST code might not be executed until later in the process, after device enumeration occurs. For system-board devices, the system BIOS should populate the Subsystem Vendor ID at power on. Add-on display adapters should provide a method for populating the Subsystem Vendor ID at the point when power is applied and the device is initialized to the state that it is ready for POST.

12. Configuration Space is correctly populated

Required

Windows places extra constraints on a few configuration registers and has uncovered some problem usage of other registers. Microsoft provides a program named Pci.exe to help debug the use of the Configuration Space. This program is available on the Microsoft FTP server, as described in the “PCI References” section at the end of this chapter.

The following items are specific requirements for the Configuration Space:

- Populate the class code register (09h) for all devices.

  Follow the base class, sub-class, and programming interface values outlined in PCI 2.1.

- Devices must not fill BARs with random values.

  See PCI 2.1 for correct usage of these registers. Notice that BARs (10, 14, 18, 1C, 20, and 24h) should return zero if they are not used, indicating that no memory or I/O space is needed.

Also, for performance reasons, it is recommended that run-time registers for PCI devices should not be placed in the Configuration Space.

13. Interrupt routing supported using ACPI

Required

The system must provide interrupt routing information using a _PRT object, as defined in Section 6.2.3 of the *Advanced Configuration and Power Interface Specification, Revision 1.0* or higher.

14. BIOS does not configure I/O systems to share PCI interrupts

Recommended

This applies to boot devices configured by the BIOS on systems based on Intel Architecture processors. The operating system should configure all other devices. For systems that will run the Windows operating system, OEMs should design the BIOS so that it does not configure the I/O systems in the PC to share PCI interrupts for boot devices. An exception exists for legacy audio devices following the configuration guidelines outlined in the white paper titled *Implementing Legacy Audio Devices on the PCI Bus*, available on the web site at http://www.intel.com/pc-supp/platform/ac97/wp/leg_pci.htm.

Windows does not support sharing an IRQ between real-mode and protected-mode code within the I/O subsystem. An example of this is an NDIS 2.0 driver (real mode) and a SCSI miniport driver (protected mode) for two PCI devices that share the same IRQ. The problem is that the IRQ needs to be reflected to real mode for the NDIS 2.0 driver to work.

However, if the IRQ is reflected to real mode, the real-mode SCSI driver (which usually is not called because Windows takes over in protected mode) might touch the hardware, which would cause the SCSI miniport to be confused. Windows resolves this problem either by switching everything to protected mode or by falling back to real mode.

15. BIOS configures boot device IRQ and writes to the interrupt line register

Required

This requirement applies to boot devices configured by the BIOS on systems based on Intel Architecture processors. Windows should configure all other devices because, after an IRQ is assigned by the system BIOS, Windows cannot change the IRQ, even if necessary. If the BIOS assigns the IRQ and Windows needs it for another device, a sharing problem occurs.

The BIOS must configure the boot device IRQ to a PCI-based IRQ and must write the IRQ into the interrupt line register 3Ch, even if the BIOS does not enable the device. This way, the operating system can still enable the device with the known IRQ at configuration time, if possible.

16. Hot swapping for any PCI device uses ACPI-based methods
Required

Windows 98 and Windows NT 5.0 support dynamic enumeration, installation, and removal of PCI devices only if there is a supported hardware insert/remove notification mechanism.

The appropriate notification mechanism is supported as a bus standard for CardBus bus controllers. For other solutions, such as those required for docking stations or other devices, the hardware insert/remove notification mechanism must be implemented as defined in Section 5.6.3 of the ACPI 1.0 specification. To properly function with the native support in the operating system, developing industry standards, such as those referred to as PCI Hot Plug and Compact PCI, must use ACPI-based methods for supporting hardware insertion and removal as defined in the ACPI 1.0 specification.

Power Management for PCI Controllers and Peripherals

This section summarizes the specific PCI power management requirements.

17. All PCI components comply with PCI Bus Power Management Interface specification
Required

The PCI bus, any PCI-to-PCI bridges on the bus, and all devices on the PCI bus must comply with *PCI Bus Power Management Interface Specification, Revision 1.0* or higher. This specification defines the OnNow device power states (D0–D3) for PCI devices. It also covers the bus functionality expected in each power state and the mechanism for signaling wake-up events over the bus. For PC 98, PCI bus implementations must support all four device power states, either the B2 or B3 bus power state, and the standard wake-up mechanism.

This specification allows the operating system to individually power manage PCI devices to conserve power, and it allows PCI peripherals to wake up the system when the need arises.

Note: It is an acceptable alternative for embedded PCI devices (on the system board) to use ACPI for power-state and wake-up control instead of the new PCI definitions, as defined in Sections 3.3, 3.4, and 7.0 of the ACPI 1.0 specification.

For add-on adapters, including AGP cards, compliance with the PCI Bus Power Management specification is required, including the Configuration Space registers and the device state (D*x*) definitions.

PCI References

The following represents some of the references, services, and tools available to help build hardware that is optimized to work with Windows operating systems.

Advanced Configuration and Power Interface Specification, Revision 1.0
http://www.teleport.com/~acpi/

"Efficient Use of PCI," Platform Architecture Labs, Intel Corporation
http://support.intel.com/oem_developer/chipsets/pci/general/pci001.htm

"IDs and Serial Numbers for Plug and Play" and other related articles
http://www.microsoft.com/hwdev/busbios/

Implementing Legacy Audio Devices on the PCI Bus
http://www.intel.com/pc-supp/platform/ac97/wp/leg_pci.htm

Microsoft testing tools, specifications, and information
E-mail: pciinfo@microsoft.com
http://www.microsoft.com/hwtest/
http://www.microsoft.com/hwdev/busbios/
ftp://ftp.microsoft.com/developr/drg/plug-and-play/pci/pci.exe

PCI Bus Power Management Interface Specification, Revision 1.0
PCI Local Bus Specification, Revision 2.1 (PCI 2.1)
PCI to PCI Bridge Specification, Revision1.0.
PCI SIG
Phone: (800) 433-5177
http://www.pcisig.com/

PCI to PCMCIA CardBus Bridge Register Description (Yenta specification)
PCMCIA
2635 North First Street, Suite 209
San Jose, CA 95134 USA
Phone: (408) 433-2273
Fax: (408) 433-9558
E-mail: office@pcmcia.org
http://www.pc-card.com/

Checklist for PCI

If a recommended feature is implemented, it must meet the PC 98 requirements for that feature as defined in this document.

1. All components comply with PCI 2.1
Required

2. System does not contain ghost cards
Required

3. System uses standard method to close BAR windows on nonsubtractive decode PCI bridges
Required

4. System supports PCI docking through a bridge connector
Recommended

5. PCI chip sets support Ultra DMA/33
Required

6. System-board bus complies with PCI 2.1
Required

7. Bus master privileges are supported for all connectors
Required

8. ISA Write Data Port address is propagated to the ISA bus at power up
Required

9. Functions in a multifunction PCI device do not share writable PCI Configuration Space bits
Required

10. Devices use PCI 2.1 Configuration Space register for Plug and Play device ID
Required

11. Device IDs include PCI 2.1 Subsystem IDs
Required

12. Configuration Space is correctly populated
Required

13. Interrupt routing supported using ACPI
Required

14. BIOS does not configure I/O systems to share PCI interrupts
Recommended

15. BIOS configures boot device IRQ and writes to the interrupt line register
Required

16. Hot swapping for any PCI device uses ACPI-based methods
Required

17. All PCI components comply with PCI Bus Power Management Interface specification
Required