PCI Device Configuration
BARs & Interrupts

Chapter 2

OBJECTIVES:
Upon completion of this topic, the participant will be able to do the following:

- Recognize Windows® Device Manager displays.
- Discuss the functions of the PCI Command register.
- Describe memory space allocation using BARs.
- Explain the use of PCI Interrupt Pin/Line Configuration Registers.

IATT Web Site: http://iatt.intel.com
PCI - Auto-configuration by Design

- Users can install a PCI peripheral device without having to manually configure jumpers/DIP switches.
  - Configuration registers are built into each component
    - Configuration registers are set up during system initialization and allow identification of the devices (SCSI, LAN, etc.)
  - These registers also allow configuration of the device's I/O addresses, memory addresses, interrupt levels.
- PCI Device configuration includes:
  - **Enabling** access to memory and/or I/O regions.
  - Allocating the **amount and location** of PCI I/O and PCI memory space a device can use.
  - Assigning **interrupts** for devices.
  - A PCI device won’t **respond** to cycles until configured.

Windows Device Manager Overview

- Device manager allows users to view hardware devices and manage those hardware devices easily.
  - Displays a list of components which you can double click to view additional information such resources it is using.
Portion of Windows DEV MGR I/O Map

Note: One I/O address associated with the AC97 device is “E800”. The allocation of Memory & I/O addresses are described later in this chapter.

AC97 I/O = E800

PCI Configuration Command Register

Enable I/O and memory accesses by setting bits 0 and 1 in the PCI Command register at offset 04H in PCI Configuration space

- CMD Register is 16 bits at offset 05-04 and provides control over device’s ability to respond to PCI accesses—e.g., allows access to I/O or memory by Device Driver.
- Bit 2 (Bus Master Enable) is only for Bus Master devices such as EIDE, SCSI, LAN.
- See the PCI spec for bit 15-3 definitions.
Making Changes to CMD Regs (1 of 2)

- The dump below of AC97 PCI configuration space shows offset 4 contains a hex "05" or binary "0101". This means bits 2 and 0 are both set—I/O & Bus Mstr are both enabled.

```
[P0]>showpci (0,1f,5) /* Device 31, Fun 5--AC97 ctlr */
Offset    F E D C B A 9 8 7 6 5 4 3 2 1 0
--------------------------------------------
0F...00: 00000000 04010005 02800005 24458086
```

Note: The I/O address associated with the AC97 device is "E800". The allocation of Memory & I/O addresses are described later in this chapter.

Making Changes to CMD Regs (2 of 2)

- The ITP Debug Tool commands below show what happens when I/O access is enabled/disabled by setting/clearing bits in the command register--offset 04h.

```
[P0]>dport (0cf8)=8000fd04;port (0cfc)
05 /* This reads 05 from AC97 cmd reg (bus mstr & I/O enabled) */
[P0]>wport (0e802) /* AC97 I/O range E800-E8FF */
8505 /* Able to read 8505 (mute & Mid Vol) from I/O addr E802 */
[P0]>/* Now change cmd reg to 00: bus mstr & I/O disabled */
[P0]>dport (0cf8)=8000fd04;port (0cfc)=00
[P0]>wport (0e802)
ffff /* No longer able to read data from I/O address E802 */
[P0]>dport (0cf8)=8000fd04;port (0cfc)=05
[P0]>/* Change cmd reg back to 05 = bus mstr & I/O enabled again */
[P0]>wport (0e802)
8505 /* Again able to read data of "8505" from I/O addr E802 */
```
The Command Register in PCI Configuration space is 16 bits starting at offset 04 and allows control of ______ and I/O space enable/disable.

Memory

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Vendor ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Command</td>
</tr>
<tr>
<td>Class Code</td>
<td>Revision ID</td>
</tr>
<tr>
<td>BIST</td>
<td>Header Type</td>
</tr>
<tr>
<td>Latency</td>
<td>Cache Line Size</td>
</tr>
<tr>
<td>Latency</td>
<td>Timer</td>
</tr>
<tr>
<td>Cache Line Size</td>
<td></td>
</tr>
<tr>
<td>Base Address Registers (32 bits each at offsets 10h-27h)</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Base Address Registers</td>
<td></td>
</tr>
</tbody>
</table>

A PCI device won't respond to ANY cycles until configured.

- **Base Address Regs (BAR)**
  - BAR is used to *allocate* the amount and location of PCI I/O and PCI Memory space a device can use.
  - Power up system software (BIOS and or O/S) is responsible for initializing BARs in a resource conflict free manner (Plug & Play)
  - Upon completion of the PNP configuration the system writes the assigned base address value for the device into the BAR. For example, an AGP card might be assigned BAR= FD000000h

Each device may use up to 6 BAR registers to identify up to 6 individual address spaces of various sizes to be used by the device.
Base Address Registers (BAR)

**BAR USE:**
Request memory and/or I/O space.

BARs provide the mechanism for mapping PCI devices into the system address space.

**Bottom bit** of Base Address Reg specifies which address space.

- Upon power-up, the BAR identifies the resources required by the device.
- BIOS and other software writes all 1’s and reads value back to determine size and alignment (see PCI spec for details)
- System software uses these addresses to access device.

**Example Memory Map - After configuration**

- **PCI Device A**: BAR = 1 GB
- **PCI Device B**: BAR = 2 GB
- **PCI Device C**: BAR = 3 GB

System Memory Space

- **Top of Memory**: (256 MB for example)
- **Physical memory (i.e., SDRAM)** does not exist above T.O.M.

PCI Bus

Bridge.
Windows Device MGR I/O Map

After Configuration:
AC97 was assigned I/O space starting at 0000E800 & 0000EF00

Windows Device MGR Memory Map

After Configuration:
AGP was assigned memory space starting at F8000000 & FC900000 & FD000000 (as well as the standard VGA Video Buffer at 000A0000--reserved mem area below Top of Memory)
PC Chipset: Functions & Devices

**ITP Dump of BARs After Configuration**

- **PCI Dev# 0 [ 0h]:** Fun# 0 DID/VID = 1a308086 [dport(0cf8)=8000003c]
  - BAR-10=f8000008, BAR-14=00000000 /* Mem Ctrlr */
- **PCI Dev# 1 [ 1h]:** Fun# 0 DID/VID = 1a318086 [dport(0cf8)=8000083c]
  - BAR-10=00000000, BAR-14=00000000 /* AGP Bridge */
- **PCI Dev# 31 [1Fh]:** Fun# 4 DID/VID = 24448086 [dport(0cf8)=8000fc3c]
  - BAR-10=00000000, BAR-14=00000000 /* USB #2 */
- **PCI Dev# 31 [1Fh]:** Fun# 5 DID/VID = 24458086 [dport(0cf8)=8000fd3c]
  - BAR-10=0000e801, BAR-14=0000ef01 /* AC97 */

Now dumping PCI bus 1.

- **PCI Dev# 0 [ 0h]:** Fun# 0 DID/VID = 011010de [dport(0cf8)=8001003c]
  - BAR-10=fd000000, BAR-14=e8000008 /* AGP card */

Now dumping PCI bus 2.

- **PCI Dev# 10 [ Ah]:** Fun# 0 DID/VID = 00109005 [dport(0cf8)=8002503c]
  - BAR-10=0000d801, BAR-14=feaff004 /* SCSI card */
- **PCI Dev# 11 [ Bh]:** Fun# 0 DID/VID = 920010b7 [dport(0cf8)=8002583c]
  - BAR-10=0000dc01, BAR-14=feafec00 /* LAN card */

**Effects of writing to AGP Memory**

- Portion of Windows desktop. The digital data representing this screen is stored in AGP video memory at A0000 and System memory at E8000000.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000000000a0000P</td>
<td>efbeefbe efbeefbe efbeefbe efbeefbe</td>
</tr>
<tr>
<td>0x000000000000a0010P</td>
<td>f79ef79e f7def7be f7def7be f7beefbe</td>
</tr>
<tr>
<td>0x000000000000a0020P</td>
<td>f79ef79e f7beefbe f7bff7bf ef9ef79f</td>
</tr>
<tr>
<td>0x000000000000a0030P</td>
<td>efbeefbe efbeefbe efbeefbe efbeefbe</td>
</tr>
<tr>
<td>0x000000000000a0040P</td>
<td>efbeefbe efbeefbe efbeefbe efbeefbe</td>
</tr>
<tr>
<td>0x000000000000a0050P</td>
<td>efbeefbe efbeefbe efbeefbe efbeefbe</td>
</tr>
</tbody>
</table>

- Desktop after writing to AGP video memory at A0000.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000000000a0000P</td>
<td>f0f0f0f0 f0f0f0f0 f0f0f0f0 f0f0f0f0</td>
</tr>
<tr>
<td>0x000000000000a0010P</td>
<td>f0f0f0f0 f0f0f0f0 f0f0f0f0 f0f0f0f0</td>
</tr>
<tr>
<td>0x000000000000a0020P</td>
<td>f0f0f0f0 f0f0f0f0 f0f0f0f0 f0f0f0f0</td>
</tr>
</tbody>
</table>

- Desktop after writing to System memory at E8000000.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000000000a0000P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
<tr>
<td>0x000000000000a0010P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
<tr>
<td>0x000000000000a0020P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
<tr>
<td>0x000000000000a0030P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
</tbody>
</table>

- "f0f0f0f0" changes part of screen to a red color.

- Desktop after writing to System memory at E8000000.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000000000000a0000P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
<tr>
<td>0x000000000000a0010P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
<tr>
<td>0x000000000000a0020P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
<tr>
<td>0x000000000000a0030P</td>
<td>33223322 33223322 33223322 33223322</td>
</tr>
</tbody>
</table>

- "33223322" changes part of red screen to a green color.

Note: Red & Green shades not visible in Grayscale handout.
The bottom bit of the Base Address Register specifies:

1: Max size of the BAR register
2: Config complete/not done
3: Memory or I/O Space
4: Amount of space required

Supported in newer O/S's (e.g., Windows 2000/XP) and supports 24 interrupts.
PCI Hardware Interrupts

- Multiple PCI devices can be assigned to same IRQ.
  - The operating system calls each interrupt handler until one of the handlers claims the interrupt.
- A typical chipset has 8 PIRQ input signals: PIRQ[H:A]#
  - PCI connector interrupt signals (INTA, B, C, D) will connect to 1 of 8 ICH interrupt inputs (PIRQA#-PIRQH#).
  - Any PCI interrupt source (onboard or from a PCI add-in card) connects to one of these PIRQ signals.
- In APIC mode, PIRQA#-PIRQH# are often routed to PCI interrupts IRQ16-IRQ23 respectively.
  - In PIC mode (legacy interrupts), the ICH can connect each PIRQ line internally to one of the IRQ signals (3-7, 9-12, 14, & 15)--each PIRQx# line has a separate Route Control register (e.g., D31:F0 offsets 60h-6Bh for PIRQA#-PIRQH#)

PCI Configuration Registers--Interrupts

- Each PCI slot has 4 pins: INTA#, INTB#, INTC#, INTD#
- INTA is used for all add-in cards that require only 1 IRQ
- INTB for the 2nd interrupt on cards that require 2 or more
- INTC & INTD: 3rd & 4th (not absolute requirements)

The interrupt line (IRQ) is read by software (O/S & drivers) from the Interrupt Line register (offset 3C) in the PCI Configuration Space of a device (3C holds hex value of IRQ).

- Interrupt Pin & Line use example:
  - Pin (3D): 01 for INTA, 02 for INTB
  - Line (3C): 10h -> IRQ16, 11h -> IRQ17

More details in next slides
Each PCI slot has 4 pins: INTA#, INTB#, INTC#, and INTD#.

True or False

Typical Interrupt Routing (from TPS)

<table>
<thead>
<tr>
<th>Typical APIC IRQs:</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI Interrupt Source</td>
<td>PIRQA</td>
<td>PIRQB</td>
<td>PIRQC</td>
<td>PIRQD</td>
<td>PIRQE</td>
<td>PIRQF</td>
<td>PIRQG</td>
<td>PIRQH</td>
</tr>
<tr>
<td>AGP connector</td>
<td>INTA</td>
<td>INTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH6 USB UHCI controller 1</td>
<td>INTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMBus controller</td>
<td>INTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH6 USB UHCI controller 2</td>
<td>INTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC ’97 ICH5 Audio</td>
<td>INTB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH5 LAN</td>
<td>INTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH6 USB UHCI controller 3</td>
<td>INTC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH6 USB UHCI controller 4</td>
<td>INTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH4 USB 2.0 HIC controller</td>
<td>INTD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI bus connector 1</td>
<td>INTD</td>
<td>INTA</td>
<td>INTB</td>
<td>INTC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI bus connector 2</td>
<td>INTC</td>
<td>INTB</td>
<td>INTA</td>
<td>INTD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- A typical chipset has 8 PIRQ input signals. PIRQ[H:A]#
- Each PCI slot has 4 pins: INTA#, INTB#, INTC#, INTD#
- INTA is used for all add-in cards that require only 1 IRQ
Typical Interrupt Map (from TPS)

<table>
<thead>
<tr>
<th>IRQ</th>
<th>System Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 *</td>
<td>Interval Timer</td>
</tr>
<tr>
<td>1 *</td>
<td>Keyboard</td>
</tr>
<tr>
<td>2 *</td>
<td>Cascade from slave PIC</td>
</tr>
<tr>
<td>3</td>
<td>COM1 – can be changed</td>
</tr>
<tr>
<td>4</td>
<td>COM2 – can be changed</td>
</tr>
<tr>
<td>5</td>
<td>LPT2 / User available</td>
</tr>
<tr>
<td>6</td>
<td>Diskette drive</td>
</tr>
<tr>
<td>7</td>
<td>LPT1 – can be changed</td>
</tr>
<tr>
<td>8</td>
<td>Real-time clock</td>
</tr>
<tr>
<td>9 *</td>
<td>ICH SM Bus</td>
</tr>
<tr>
<td>10 (0ah)</td>
<td>User available</td>
</tr>
<tr>
<td>11 (0bh)</td>
<td>User available</td>
</tr>
<tr>
<td>12 (0ch)</td>
<td>Mouse / User available</td>
</tr>
<tr>
<td>13 (0dh)</td>
<td>Math Coprocessor</td>
</tr>
<tr>
<td>14 (0eh)</td>
<td>Pri IDE / User available</td>
</tr>
<tr>
<td>15 (0ff)</td>
<td>Sec IDE / User available</td>
</tr>
<tr>
<td>16-23 Available in APIC mode only</td>
<td></td>
</tr>
<tr>
<td>17 (11h)</td>
<td>AC97 / avail (PIRQB)</td>
</tr>
<tr>
<td>18 (12h)</td>
<td>ICH USB Ctrl 3 (PIRC)</td>
</tr>
<tr>
<td>19 (13h)</td>
<td>ICH USB Ctrl 2 (PIRQD)</td>
</tr>
<tr>
<td>20 (14h)</td>
<td>ICH LAN (opt) (PIRQE)</td>
</tr>
<tr>
<td>21 (15h)</td>
<td>User available (PIRQF)</td>
</tr>
<tr>
<td>22 (16h)</td>
<td>User available (PIRQG)</td>
</tr>
<tr>
<td>23 (17h)</td>
<td>ICH USB2 EHCI (PIRQH)</td>
</tr>
</tbody>
</table>

* = Reserved
16-23 avail in APIC mode only

Device MGR IRQs (Win 2K w/ PIC)

Note that with the standard ISA compatible 8259 PICs (Programmable Interrupt Controllers), many PCI devices share interrupt 9

Note – NOT USING APIC
Note that with the APIC (Advanced Programmable Interrupt Controller), PCI devices do not need to share interrupt 9, and most are assigned their own interrupt.

Device MGR IRQs (Win XP w/ APIC)

- **PCI Dev# 0 [0h]: Fun# 0 DID/VID = 1a308086 [dport(0cf8)=8000003c]
  Intr Pin=00:No INT, Intr Line=0:0h /* Mem Ctrlr NO INTERRUPTS USED */
- **PCI Dev# 31 [1Fh]: Fun# 2 DID/VID = 24428086 [dport(0cf8)=8000fa3c]
  Intr Pin=04:INTD, Intr Line=19:13h /* USB #1 - PCI Interrupt 19 */
- **PCI Dev# 31 [1Fh]: Fun# 3 DID/VID = 24438086 [dport(0cf8)=8000f33c]
  Intr Pin=02:INTB, Intr Line=10:0ah /* SMBus Ctrlr - PCI Interrupt 10 */
- **PCI Dev# 31 [1Fh]: Fun# 4 DID/VID = 24448086 [dport(0cf8)=8000fc3c]
  Intr Pin=03:INTC, Intr Line=23:17h /* USB #2 - PCI Interrupt 23 */
- **PCI Dev# 31 [1Fh]: Fun# 5 DID/VID = 24458086 [dport(0cf8)=8000fd3c]
  Intr Pin=02:INTB, Intr Line=17:11h /* AC97 - PCI Interrupt 17 */

Now dumping PCI bus 1.

Note: Partial list – not all devices shown

ITP Dump of Interrupt Pin/Line (APIC)

- **PCI Dev# 0 [0h]: Fun# 0 DID/VID = 011010de [dport(0cf8)=8000003c]
  Intr Pin=01:INTA, Intr Line=16:10h /* AGP card - PCI Interrupt 16 */

Now dumping PCI bus 2.

- **PCI Dev# 10 [Ah]: Fun# 0 DID/VID = 00109005 [dport(0cf8)=8002503c]
  Intr Pin=01:INTA, Intr Line=22:16h /* SCSI card - PCI Interrupt 22 */
  /* SCSI card in PCI slot 2--this would be IRQ21 if card in PCI slot 1 */
- **PCI Dev# 11 [Bh]: Fun# 0 DID/VID = 920010b7 [dport(0cf8)=8002583c]
  Intr Pin=01:INTA, Intr Line=23:17h /* LAN card - PCI Interrupt 23 */
SYSTEM BUS VS. APIC BUS

- PCI Message signaled interrupts (MSI) are available on newer systems with (x)APIC architecture.
  - The xAPIC architecture (Pentium 4 and Xeon) is an extension of the P6 family APIC architecture.
    - \((x) = \text{Extensions}--\text{extra features beyond standard I/O APIC.}\)
    - \(x\text{I/OAPIC sends interrupt requests to CPUs on system bus.}\)
    - I/O APIC & CPU APICs communicate with 3-wire APIC bus
      - Standard APIC bus messages are invisible to software
  - Message signaled interrupts allow devices to request interrupt service via a standard PCI transaction (S/W)
    - Eliminates need for interrupt traces and sharing IRQs
    - Number of MSIs is virtually unlimited
    - Traditionally, when a PCI device signals an interrupt it grounds one of its H/W pins (INTx\# where \(x\) is \(\{A, B, C, D\}\)).

Chapter 3 Quiz

1) The Command Register is 16 bits at offset 05-04 in PCI Configuration space and provides control over:
   A) Bus Master Enable
   B) Memory Space Enable
   C) I/O Space Enable
   D) All of the above

2) The _________ is used to allocate the amount and location of PCI I/O and PCI Memory space a device can use.
   A) RR (Resource Reg)  C) BAR (Base Address Reg)
   B) IOR (IO Enable Reg)  D) MAR (Mem Allocation Reg)

3) The hex value of the IRQ is read by software from the Interrupt Line register (offset 3C) in the PCI Configuration Space of a device. (True / False)
**REVIEW & SUMMARY**

- **Windows ® Device Manager display.**
  - Device manager allows users to view hardware devices and manage those hardware devices easily.
  - Displays IRQs, DMA, and Memory & I/O maps
- **Functions of the PCI Command register**
  - CMD Reg (16 bits at offset 05-04) and provides control over device’s ability to respond to PCI accesses
  - Enable I/O & memory accesses by setting bits 0 and 1
- **Memory space allocation using BARs**
  - **Base Address Registers** are used to allocate amount & location of PCI I/O and Memory space a device can use.
  - Power up system software (BIOS and or O/S) is responsible for initializing BARs in a resource conflict free manner.

**REVIEW & SUMMARY**

- **Use of PCI Interrupt Pin/Line Configuration Registers**
  - PCI connector interrupts signals (INTA, B, C, D) will connect to 1 of 8 ICH interrupt inputs (PIRQA#-PIRQH#).
  - **Interrupt Pin at offset 3D; Interrupt Line at offset 3C**
    - Interrupt Pin (3D): 01 for INTA; 02 for INTB
    - Interrupt Line (3C): 17 -> IRQ17, 23 -> IRQ23
  - **With the APIC (24 IRQs), most devices are assigned their own interrupts (e.g., IRQ16-23)**
    - With standard ISA compatible 8259 PICs (15 IRQs), many devices share interrupts (for example IRQ 9).