iHAWK Optimization Guide

Adding a new board to your iHawk system?

Looking to improve overall system performance?

→ *Go to Chapter 2 for a quick reference to recommended board placements on individual iHawk systems.*

Need to understand more about the PCI subsystem and how it affects system performance?

→ *Go to Chapter 3 for an in-depth discussion of the factors that affect performance and tools you can use.*

concurrent

0898011-110
April 2008

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent makes no warranties, expressed or implied, concerning the information contained in this document.

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive, Pompano Beach, Florida, 33069. Mark the envelope **"Attention: Publications Department**." This publication may not be reproduced for any other reason in any form without written permission of the publisher.

Printed in U. S. A.

Revision History	Date	Level
Original Release	May 2005	000
Minor Update	July 2006	010
Restructure	May 2007	100
Update	April 2008	110

Preface

Scope of Manual

This manual is directed toward those responsible for installation, configuration and administration of Concurrent iHawk computer systems.

Even though this guide represents a limited subset of architectures offered by Concurrent Computer Corporation, these guidelines may be helpful in improving performance on other PCI-based systems running one of Concurrent's real-time Linux operating systems.

Structure of Manual

This guide consists of the following sections:

- Chapter 1, *Overview*, explains the purpose of this guide and how to use it.
- Chapter 2, *Configuration Recommendations*, provides PCI slot and bus layouts and recommendations for configuring PCI boards based on the IRQ management scheme used on each of the iHawk systems.
- Chapter 3, *Performance Factors in Configuring iHawks*, provides in-depth discussions of how PCI board configuration and IRQ management can be manipulated to improve system performance.
- The *Glossary* provides definitions of terms used throughout this guide.
- The *Index* contains an alphabetical reference to key terms and concepts and the pages where they occur in the text.

Related Publications

Concurrent Documentation	**Pub No.**
RedHawk Linux Operating System	
RedHawk Linux Release Notes	0898003
RedHawk Linux User's Guide	0898004
SUSE Linux Enterprise Real Time Operating System	
SUSE Linux Enterprise Real Time 10 Release Notes	0898206
SUSE Linux Enterprise Real Time User's Guide	0898204
RCIM PCI Form Factor	
Real-Time Clock and Interrupt Module (RCIM) User's Guide	0898007

Contents

Chapter 3 Performance Factors in Configuring iHawks

1
Overview

1
Overview

How to Use This Guide

Whether you are initially configuring an iHawk system, installing an add-on board or tuning system performance, use the recommendations in this guide to help determine the most desirable and optimal configuration possible for your system.

How PCI slots are populated, the types and characteristics of PCI buses on a system and IRQ priority management schemes all affect system performance. In general you should consider two things:

- IRQ processing
- Bus contention

This guide serves to document these issues and attempts to suggest methods you can use to find a configuration that does not negatively impact your particular application.

As you read through this guide, understand that you will not find the perfect configuration for your system neatly contained in a single table. **The tables in Chapter 2 of this guide are provided as an EXAMPLE ONLY in order to illustrate concepts.** Variations in kernel releases and BIOS revisions make it impossible to provide a single reference for every system.

Go to Chapter 3 to understand the impact of IRQ processing and bus contention within the context of your application. Tools provided in Concurrent's real-time Linux operating systems for troubleshooting your system's PCI assignments and values are also described. For details beyond the scope of this manual, refer to the manufacturer's system guides.

NOTE

You may find differences between spec sheets, diagrams printed on the case of your system and the contents of this guide. Use the **`lsirq(8)`** and **`lspci(8)`** tools together with this guide to find the correct bus layout and IRQ assignments for your system.

General Factors in IRQ Processing and Bus Contention

Here are some basic facts to keep in mind as you use this guide.

Factors that relate to IRQ processing include:

- IRQs are prioritized at the hardware level.
- Hardware interrupts preempt software interrupts.
- All interrupts preempt all software processes.
- PCI IRQs can be shared.
- IRQs are assigned by the kernel based on hardware resources and device discovery.
- IRQ assignment is complex and somewhat unpredictable.
- Reconfiguring hardware may or may not affect IRQ assignments.
- The trend is that slower buses get lower priority IRQs.

Factors that relate to PCI bus contention include:

- Devices on the same PCI bus can only run at the speed of the slowest device.
- Only one device at a time can access the PCI bus.
- PCI latency describes the number of bus cycles a device is allocated per request.
- Different devices have different PCI bus latencies.
- High bandwidth devices have higher PCI bus latencies.
- PCI buses connect to the system bus through a bridge chip.
- There can be multiple independent PCI buses in a system.
- Devices are assigned a logical slot address.
- More than one logical slot address can be assigned to a phyiscal slot.

Chapter 3 discusses these factors in greater length.

Here are some example scenarios for configuring Concurrent's iHawk systems with the Real-time Clock and Interrupt Module (RCIM), or any other real-time device:

The ideal configuration:

- The RCIM has the top priority IRQ.
- The IRQ is unshared.
- The RCIM is on its own bus.

The worst configuration:

- The RCIM has the lowest priority IRQ.
- The IRQ is shared with several busy devices.
- The RCIM shares the bus with several busy devices with high PCI latencies.

In reality, you may be forced to choose a compromise:

- The RCIM does not have the top priority IRQ.
- The IRQ is shared with inactive devices.
- The RCIM shares the bus with devices that don't hog the bus.

1
Configuration Recommendations

2
Configuration Recommendations

This chapter provides PCI board placement recommendations for iHawk models.

The tables in this chapter are provided as an EXAMPLE ONLY in order to illustrate concepts discussed in Chapter 1 and Chapter 3.

How to Use the IRQ Priority Management Tables

This section explains how to understand and utilize the information provided in the IRQ Priority Management tables in this chapter for iHawk models.

Quick Reference

With this slot population, devices installed in PCI slot(s) listed under Slot Priority 1 receive the highest slot priority; however, an embedded device might receive a higher device priority.

Configuration (1)	Slot Priority			
	1	2	3	4
	slot numbers			
RCIM only	4	n/a		
4 PCI-X slots filled	4	5	3	2
Legacy slot 1 + 1 PCI-X slot filled	2-5	1	n/a	
Legacy slot 1 + 2 PCI-X slots filled	2	4/5	1	n/a
Note that one scheme that favors priority may not favor bandwidth.	3	2/4/5	1	

(1) With USB enabled in the BIOS, all slots are forced to a device priority lower than the embedded devices.

Special notes apply to the individual system.

More than one slot can receive the same priority. These notations indicate "slots 2 through 5" and "slots 2, 4 and 5."

Slot Priority vs. Device Priority vs. System Priority

The IRQ Priority Management tables represent the priority ranking of the physical PCI adapter slots without regard to embedded devices. Embedded devices may be assigned priorities that fall above or below a given physical slot. Because of this, we use the terms "slot priority," "device priority" and "system priority."

slot priority — includes all maskable hardware interrupts belonging only to PCI cards installed in the system.

device priority — includes all maskable hardware interrupts: physical slots and embedded devices.

system priority — includes all maskable and non-maskable hardware interrupts

As an example, the following table presents a hypothetical system.

Device	Device Priority	Slot Priority
Embedded NIC2	1	n/a
RCIM card	2	1
Embedded NIC1	3	n/a
PCI card A	4	2
PCI card B	5	3
PCI card C	6	4
Embedded SCSI	7	n/a

On this system, the RCIM has the highest slot priority, but receives device priority 2 behind the Embedded NIC2. If NIC2 is disabled in the BIOS, the RCIM will then receive the highest device priority. If this NIC2 had been the only ethernet controller in the system, disabling it in the BIOS and installing another NIC in a PCI slot with a slot priority 2 or lower would assign the RCIM as device priority 1 and still retain the NIC2 but with a lower device priority. Use **`lsirq(8)`** to view the device priority assignments (see Chapter 3 or the man page).

In addition to the PCI devices and embedded devices competing for hardware priorities in the system, there are non-maskable hardware interrupts. In a display of **`/proc/interrupts`**, everything listed in the bottom section starting with NMI represents a non-maskable hardware interrupt; for example:

```
$ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
  0:         51          0    2448698          0    IO-APIC-edge  timer
  3:          0          0          0          0    IO-APIC-edge  KGDB-stub
  4:        284          0          0          0    IO-APIC-edge  serial
  9:          0          0          0          0   IO-APIC-level  acpi
 14:          0          0         21          1    IO-APIC-edge  ide0
 50:      16943          0          0          0   IO-APIC-level  eth0
177:          0          0          0          0   IO-APIC-level  uhci_hcd
185:          0          0          0          0   IO-APIC-level  uhci_hcd
193:          0          0          0          0   IO-APIC-level  uhci_hcd
201:          0          0          0         16   IO-APIC-level  ehci_hcd
209:         29          0          0          0   IO-APIC-level  ioc0
217:          0       3861          0          0   IO-APIC-level  ioc1
225:          0          0          0          0   IO-APIC-level  btp
233:          0          0          0          0   IO-APIC-level  rcim

NMI:        285        265        194        107   Non-maskable interrupts
LOC:    2448418    2448383    2448431    2448353   Local interrupts
RES:         93         77         81        106   Rescheduling interrupts
CAL:        112        113        116         58   function call interrupts
TLB:         95        151         97         99   TLB shootdowns
TRM:          0          0          0          0   Thermal event interrupts
```

```
SPU:          0          0          0          0   Spurious interrupts
ERR:          0          0          0          0   Error interrupts
MIS:          0          0          0          0   APIC errata fixups
```

Non-maskable interrupts are not configurable and will always have higher system priority over maskable interrupts.

Installing Add-on Cards

The diagrams below show the most common slots found on iHawk systems. Use these diagrams as a guide to find the appropriate slot for adding cards to your system.

64-bit 133/100/66/33 MHz PCI/PCI-X Slot — 3.3 volts

64-bit 33 MHz PCI Slot — 5 volts

32-bit 33 MHz PCI Slot — 3.3 volts

32-bit 33 MHz PCI Slot — 5 volts

PCI Express x16 slot

Insert the card firmly into the appropriate slot. Do not force the card into the slot. If the card does not seat properly, try another slot or return the faulty card.

iHawk Configurations

In the rest of this chapter, the following information is given for each iHawk model:

- **Specifications** provides the iHawk model number, manufacturer, processor type, number and types of PCI slots, number of PCI buses and a list of the embedded devices.
- **PCI Bus and Slot Layout** provides PCI slot/bus relationships, bus speed/bandwith, slot voltages and PCI bus lengths.

  Note that you may find differences between spec sheets, diagrams printed on the case of your system and the contents of these tables. Use the **`lsirq(8)`** and **`lspci(8)`** tools together with this information.
- **IRQ Priority Management** provides an easy-to-read table for determining which slots to place boards in order to achieve the best possible performance based on how the system prioritizes and services interrupt requests.

iHawk Model HR430

Specifications

Manufacturer/Model	**Newisys® 4300**
Processor	**AMD Opteron™**
Total number of PCI slots	7
PCI/PCI-X	7
PCI Express	0
Hot-plugs	4
Number of PCI buses	6
Embedded devices	2 Broadcom Tigon3 BCM95703A30 Gigabit Ethernet controllers 1 LSI Logic 53C1030 Fusion-MPT U320 SCSI controller 1 AMD8111 IDE controller 1 SVGA Trident Blade3D controller 2 AMD8111 USB controllers

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HR430 system. The slots are numbered on the system case.

Slot Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length	Shares PCI bus with
1 (1)	PCI	3.3	66MHz	half-length 64-bit	embedded devices
2	PCI-X	3.3	100MHz	half-length 64-bit	slot 3
3	PCI-X	3.3	100MHz	half-length 64-bit	slot 2
4	PCI-X hot-plug	3.3	133MHz	full-length 64-bit	not shared
5	PCI-X hot-plug	3.3	133MHz	full-length 64-bit	not shared
6	PCI-X hot-plug	3.3	133MHz	full-length 64-bit	not shared
7	PCI-X hot-plug	3.3	133MHz	full-length 64-bit	not shared

(1) Embedded devices are slowed to the speed of the board installed in slot 1. Use slot 1 only if no other options are available.

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HR430 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration (1) (2) (3)	**Slot Priority**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
	slot numbers						
RCIM only	5	n/a					
Only 64-bit PCI-X 133 MHz slots filled (USB on or off)	5	4	7	6	n/a		
Only 64-bit PCI-X 100 MHz slots filled (USB on or off)	3	2	n/a				
5 64-bit PCI-X slots filled (USB off)	5	4	7	6	3/2	n/a	
All 64-bit PCI-X slots filled (USB off)	4	7	6	3	2	5	n/a
All slots filled (USB off)	7	4	3	2	1	5	6
Legacy slot 1 + 5 PCI-X slots filled (USB off)	4	7	6	3/2	1	5	n/a
Legacy slot 1 + all PCI-X 133 MHz slots filled (USB off)	5	4	7	6	1	n/a	
Legacy slot 1+ all PCI-X 100 MHz slots filled (USB on or off)	3	2	1	n/a			

(1) IRQ assignments are easier to manage with USB disabled in the BIOS. Disable USB if not needed.
(2) IRQ settings cannot be manually controlled in the BIOS on this system.
(3) Embedded ethernet controllers cannot be disabled in the BIOS on this system.

iHawk Model HR210

Specifications

Manufacturer/Model	**Newisys® 2100**
Processor	**AMD Opteron™**
Total number of PCI slots	2
PCI/PCI-X	2
PCI Express	0
Hot-plugs	0
Number of PCI buses	2
Embedded devices	2 Broadcom Tigon3 BCM5703x Gigabit Ethernet controllers 1 LSI Logic 53C1030 Fusion-MPT Dual Channel U320 SCSI controller 1 SVGA Trident Blade3D controller 2 AMD8111 USB controllers

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HR210 system. The slots are numbered on the main board next to each slot.

Slot Number (1)	**PCI Type**	**Voltage**	**Maximum PCI Bus Speed**	**PCI Bus Length**	**Shares PCI bus with**
1	PCI-X	3.3	66 MHz	full-length 64-bit	not shared
2	PCI-X	3.3	133 MHz	full-length 64-bit	not shared

(1) These slot numbers reflect the numbering provided on the hardware. The BIOS uses “slot 0” and “slot 1.”

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HR210 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration (1)	**Slot Priority**	
	1	**2**
	slot numbers	
RCIM only	2/1	n/a
1 slot filled	2/1	n/a
2 slots filled	2	1

(1) Managing IRQ priorities is usually more straightforward with USB disabled in the BIOS. This is especially true with multi-IRQ boards, which may not assign the IRQ priorities in a truly consecutive order (e.g., device priority 2, 3, 4, 10)

iHawk Model HQ685

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 6850**
Processor	**Intel® Xeon™**
Total number of PCI slots	7
PCI/PCI-X	3
PCI Express	4
Hot-plugs	4
Number of PCI buses	6
Embedded devices	1 VGA ATI Radeon/16MB SRAM 2 Broadcom BCM5704 Dual Gigabit Ethernet controllers 1 LSI Logic 53C1030 Dual Integrated PCI Ultra320 LVD SCSI controller 1 Intel 82801EB/ER (ICH5/ICH5R) USB2 EHCI controller 3 Intel 82801EB/ER (ICH5/ICH5R) USB UHCI #1

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ685 system.

Slot Number	Bus Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length
1	1	PCI Express hot-plug	3.3	40 Gbps	x8 lane width
2	2	PCI-X	3.3	133 MHz	full-length 64-bit
3	3	PCI Express hot-plug	3.3	20 Gbps	x4 lane width
4	4	PCI Express hot-plug	3.3	20 Gbps	x4 lane width
5	5	PCI Express hot-plug	3.3	20 Gbps	x4 lane width
6	6	PCI-X	3.3	100 MHz	full-length 64-bit
7	6	PCI-X	3.3	100 MHz	full-length 64-bit

(1) Embedded USB, SCSI and Ethernet cannot be disabled unless a corresponding PCI adapter is installed.

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ685 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration	Slot Priority (1)		
	1	2	3
	slot numbers		
RCIM only	6/7/2	n/a	
1 PCI-X slot filled	6/7/2	n/a	
2 PCI-X slots filled	6	7	n/a
	2	6/7	
3 PCI-X slots filled	2	7	6

(1) PCI Express cards receive priorities in the mid level range and have less impact on PCI-X slot priorities than other PCI-X cards.

iHawk Model HQ680

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 6800**
Processor	**Intel® Xeon™**
Total number of PCI slots	7
PCI/PCI-X	3
PCI Express	4
Hot-plugs	0
Number of PCI buses	6
Embedded devices	1 VGA ATI Radeon/ 16MB SRAM 2 Broadcom BCM5704 Dual Gigabit Ethernet controllers 1 LSI Logic 53C1030 Dual Integrated PCI Ultra320 LVD SCSI controller 1 Intel 82801EB/ER (ICH5/ICH5R) USB2 EHCI controller 3 Intel 82801EB/ER (ICH5/ICH5R) USB UHCI #1

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ680 system.

Slot Number	Bus Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length
1	1	PCI Express	3.3	40 Gbps	x8 lane width
2	2	PCI-X	3.3	133 MHz	full-length 64-bit
3	3	PCI Express	3.3	20 Gbps	x4 lane width
4	4	PCI Express	3.3	20 Gbps	x4 lane width
5	5	PCI Express	3.3	20 Gbps	x4 lane width
6	6	PCI-X	3.3	100 MHz	full-length 64-bit
7	6	PCI-X	3.3	100 MHz	full-length 64-bit

(1) Embedded USB, SCSI and Ethernet cannot be disabled unless a corresponding PCI adapter is installed.

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ680 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration	Slot Priority (1)		
	1	2	3
	slot numbers		
RCIM only	6/7/2	n/a	
1 PCI-X slot filled	6/7/2	n/a	
2 PCI-X slots filled	6	7	n/a
	2	6/7	
3 PCI-X slots filled	2	7	6

(1) PCI Express cards receive priorities in the mid level range and have less impact on PCI-X slot priorities than other PCI-X cards.

iHawk Model HQ665

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 6650**
Processor	**Intel® Xeon™**
Total number of PCI slots	8
PCI/PCI-X	8
PCI Express	0
Hot-plugs	0
Number of PCI buses	6
Embedded devices	2 Broadcom Tigon 3 BCM 5700 Gigabit Ethernet controllers 1 Adaptec AIC7892 U160 SCSI controller 1 Serverworks CSB5 IDE controller 1 Serverworks OSB4/CSB5 OHCI USB controller

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ665 system. Slot numbers are provided on the main board adjacent to each PCI slot and on the outside of the case.

Slot Number	Bus Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length
1 (1)	1	PCI 2.2	5	33 MHz	half-length 32-bit
2 (2)	3	PCI-X	3.3	100 MHz	full-length 64-bit
3 (2)	3	PCI-X	3.3	100 MHz	full-length 64-bit
4	4	PCI-X	3.3	100 MHz	full-length 64-bit
5	4	PCI-X	3.3	100 MHz	full-length 64-bit
6	5	PCI-X	3.3	100 MHz	full-length 64-bit
7	5	PCI-X	3.3	100 MHz	full-length 64-bit
8	6	PCI-X	3.3	100 MHz	full-length 64-bit

(1) Remote access cards must be installed in slot 1.

(2) RAID controllers for internal hard drives must be installed in slot 2 or slot 3.

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ665 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in a slot receiving Priority 1.

Configuration	Slot Priority (1)							
	1	2	3	4	5	6	7	8
	slot numbers							
RCIM only	8-4	n/a						
1 PCI-X slot filled	8-4	n/a						
2 PCI-X slots filled	8	7/6	n/a					
	7	6						
	5	4						
	3	2						
3 PCI-X slots filled	7	8	6	n/a				
	5	4	3/2					
4 PCI-X slots filled	7	8	5/4	6	n/a			
	7	8	6	3/2				
	5	4	3	2				
5 PCI-X slots filled	8	5	4	6	7	n/a		
	7	6	3	2	8			
	8	5/4	6	3/2	7			
6 PCI-X slots filled	5	4	6	3/2	7	8	n/a	
All PCI-X slots filled	4	6	3	2	7	8	5	n/a
Legacy slot 1 + 1 PCI-X slot	8-2	1	n/a					
Legacy slot 1 + 2 PCI-X slots	7	8	1	n/a				
	8/7	6	1					
	5/4	3/2	1					
Legacy slot 1 + 3 PCI-X slots	7	8	6	1	n/a			
	5	4	3/2	1				
Legacy slot 1 + 4 PCI-X slots	8	5/4	6	1	7	n/a		
	8	6	3/2	1	7			
	4	3	2	1	5			
Legacy slot 1 + 5 PCI-X slots	5	4	6	1	7	8	n/a	
	5	6	3	1	7	8		
	6	3	2	1	7	8		
Legacy slot 1 + 6 PCI-X slots	4	6	3/2	1	7	8	5	n/a
	6	3	2	1	7	8	5/4	
Legacy slot 1 + 7 PCI-X slots	6	3	2	1	7	8	5	4

(1) Embedded devices sometimes receive higher IRQ priorities than PCI slots. Slot priority may be boosted by disabling embedded devices, trying different slot populations, or a combination of both.

iHawk Model HQ660

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 6600**
Processor	**Intel® Xeon™**
Total number of PCI slots	11
PCI/PCI-X	11
PCI Express	0
Hot-plugs	0
Number of PCI buses	7
Embedded devices	2 Broadcom Tigon3 BCM9500A6 Gigabit Ethernet controllers 2 Adaptec AIC7892 SCSI controllers 1 Serverworks CSB5 IDE controller 1 Serverworks OSB4/CSB5 OHCI USB controller

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ660 system. Slot numbers are marked on the main board adjacent to each PCI slot and on the outside of the case.

Slot Number	Bus Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length
1 (1)	1	PCI 2.2	5	33 MHz	half-length 32-bit
2 (2)	3	PCI-X	3.3	100 MHz	full-length 64-bit
3 (2)	3	PCI-X	3.3	100 MHz	full-length 64-bit
4	4	PCI-X	3.3	100 MHz	full-length 64-bit
5	4	PCI-X	3.3	100 MHz	full-length 64-bit
6	5	PCI-X	3.3	100 MHz	full-length 64-bit
7	5	PCI-X	3.3	100 MHz	full-length 64-bit
8	6	PCI-X	3.3	100 MHz	full-length 64-bit
9	6	PCI-X	3.3	100 MHz	full-length 64-bit
10	7	PCI-X	3.3	100 MHz	full-length 64-bit
11	7	PCI-X	3.3	100 MHz	full-length 64-bit

(1) Remote access cards must be installed in slot 1.

(2) RAID controllers for internal hard drives must be installed in slot 2 or slot 3.

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ660 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration	Slot Priority										
	1	2	3	4	5	6	7	8	9	10	11
	slot numbers										
RCIM only	11/10	n/a									
1 PCI-X slot filled on separate bus (NIC2 on/off)	11/10	n/a									
2 PCI-X slots filled on separate buses (NIC2 on/off)	9/8	11/10	n/a								
3 PCI-X slots filled on separate buses (NIC2 on/off)	9/8	11/10	7/6	n/a							
4 PCI-X slots filled on separate buses (NIC2 on)	11/10	7/6	5/4	9/8	n/a						
4 PCI-X slots filled on separate buses (NIC2 off)	9/8	11/10	5/4	7/6	n/a						
5 PCI-X slots filled on separate buses (NIC2 on)	11/10	5/4	7/6	3/2	9/8	n/a					
5 PCI-X slots filled on separate buses (NIC2 off)	9/8	11/10	5/4	7/6	3/2	n/a					
10 PCI-X slots filled (NIC2 on)	7	6	3	2	9	8	11	10	5	4	n/a
10 PCI-X slots filled (NIC2 off)	4	7	6	3	2	9	8	11	10	5	n/a
Legacy slot 1 with 1 PCI-X slot on separate bus (NIC2 on/off)	11/10	1	n/a								
Legacy slot 1 with 2 PCI-X slots on separate buses (NIC2 on/off)	9/8	11/10	1	n/a							
Legacy slot 1 with 3 PCI-X slots on separate buses (NIC2 on/off)	9/8	11/10	7/6	1	n/a						
Legacy slot 1 with 4 PCI-X slots on separate buses (NIC2 on)	11/10	7/6	5/4	1	9/8	n/a					
Legacy slot 1 with 4 PCI-X slots on separate buses (NIC2 off)	9/8	11/10	5/4	7/6	1	n/a					
Legacy slot 1 with 5 PCI-X slots on separate buses (NIC2 on)	5/4	7/6	3/2	1	9/8	11/10	n/a				
Legacy slot 1 with 5 PCI-X slots on separate buses (NIC2 off)	11/10	5/4	7/6	3/2	1	9/8	n/a				
Legacy slot 1 with 10 PCI-X slots (NIC2 on)	6	3	2	1	9	8	11	10	5	4	7
Legacy slot 1 with 10 PCI-X slots (NIC2 off)	7	6	3	2	1	9	8	11	10	5	4

iHawk Model HQ460

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 4600**
Processor	**Intel® Xeon™**
Total number of PCI slots	7
PCI/PCI-X	7
PCI Express	0
Hot-plugs	0
Number of PCI buses	4
Embedded devices	1 Intel 82559 10/100 Ethernet controller 1 Broadcom Tigon3 Gigabit Ethernet controller 1 Adaptec AIC7899 Dual U160 SCSI controller 1 Adaptec AIC7890/91 Ultra2 SCSI controller 1 Serverworks CSB5 IDE controller 1 Serverworks OSB4/CSB5 OHCI USB controller

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ460 system. Slot numbers are provided on the main board adjacent to each PCI slot and on the outside of the case.

Slot Number	Bus Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length
1 (1)	0	PCI 2.2	5	33 MHz	half-length 32-bit
2	1	PCI-X	3.3	100 MHz	full-length 64-bit
3	1	PCI-X	3.3	100 MHz	full-length 64-bit
4	2	PCI-X	3.3	100 MHz	full-length 64-bit
5	2	PCI-X	3.3	100 MHz	full-length 64-bit
6	3	PCI-X	3.3	100 MHz	full-length 64-bit
7	3	PCI-X	3.3	100 MHz	full-length 64-bit

(1) Slot 1 is on its own bus and only appears in the BIOS when a card is installed.

IRQ Priority Management

This table shows a variety of configuration options on the iHawk HQ460 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration	Slot Priority (1)						
	1	2	3	4	5	6	7
	slot numbers						
RCIM only	any	n/a					
1 PCI-X slot filled	any	n/a					
2 PCI-X slots filled	7	6	n/a				
	6	5					
	4	2					
3 PCI-X slots filled	6	5/4/3/2	7	n/a			
	5	4/3/2	7				
	4	3/2	7				
	3	2	7				
	5	4/3/2	6				
	4	3/2	6				
	3	2	6				
4 PCI-X slots filled	5	4	7	6	n/a		
	4	3	6	5			
	3	2	5	4			
	3	2	7	6			
5 PCI-X slots filled	4	3	7	6	5	n/a	
	3	2	6	5	4		
All PCI-X slots filled	3	2	7	6	5	4	n/a
Legacy slot 1 + 1 PCI-X slot	any	1	n/a				
Legacy slot 1 + 2 PCI-X slots	2	1	4	n/a			
	3	1	5				
	4	1	6				
	5	1	7				
Legacy slot 1 + 3 PCI-X slots	2	1	4	3	n/a		
	2	1	6	4			
	3	1	7	5			
	4	1	6	5			
Legacy slot 1 + 4 PCI-X slots	2	1	5	4	3	n/a	
	2	1	7	6	3		
	4	1	7	6	5		
	3	1	7	6	4		
	3	1	6	5	4		
Legacy slot 1 + 5 PCI-X slots	2	1	6	5	4	3	n/a
Legacy slot 1 + 6 PCI-X slots	2	1	7	6	5	4	3

(1) Embedded devices sometimes receive higher IRQ priorities than PCI slots. Slot priority may be boosted by disabling embedded devices, trying different slot populations, or a combination of both.

iHawk Model HQ285

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 2850**
Processor	**Intel® Xeon™**
Total number of PCI slots	3
PCI/PCI-X	3
PCI Express	0
Hot-plugs	0
Number of PCI buses	2
Embedded devices	2 Tabor 82541EI Gigabit Ethernet controllers 1 Dual Channel LSI53C1030 U320 SCSI controller 1 Serverworks ICH5 IDE controller 1 VGA ATI Radeon 7000-M 1 USB hub with 4 controllers

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ285 system. The slots are numbered on the system case.

Slot Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length	Shares PCI bus with
1	PCI-X	3.3	133 MHz	full-length 64-bit	slot 2
2	PCI-X	3.3	133 MHz	full-length 64-bit	slot 1
3	PCI-X	3.3	133 MHz	full-length 64-bit	not shared

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ285 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration (1) (2)	**Slot Priority**		
	1	**2**	**3**
	slot numbers		
RCIM only	2	n/a	
2 slots filled (USB off)	2	1/3	n/a
	1	3	
3 slots filled (USB off)	2	1	3

(1) With USB enabled in the BIOS, all slots are forced to a priority lower than the embedded devices.

(2) With USB disabled in the BIOS, all slots receive a priority higher than the embedded devices, with the exception of the VGA controller. The embedded VGA controller will receive the highest priority with USB disabled, except when all slots are filled, in which case, slot 2 will receive the highest priority.

iHawk Model HQ280

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 2800**
Processor	**Intel® Xeon™**
Total number of PCI slots	7
PCI/PCI-X	5
PCI Express	2
Hot-plugs	2
Number of PCI buses	6
Embedded devices	2 Intel 82541GI/PI Gigabit Ethernet controllers 2 LSI Logic 53C1030 U320 SCSI controllers 1 Intel ICH5 IDE controller 1 ATI Radeon 7000/VE VGA controller 4 Intel 82801EB/ER (ICH5/ICH5R) USB controllers

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ280 system. The slots are numbered on the inside of the case on the top cover.

Slot Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length	Shares PCI bus with
1	PCI	5	33 MHz	half-length 32-bit	shared vectors
2	PCI-X	3.3	133 MHz	full-length 64-bit	slot 3
3	PCI-X	3.3	133 MHz	full-length 64-bit	slot 2
4	PCI-X	3.3	133 MHz	full-length 64-bit	not shared
5	PCI-X	3.3	133 MHz	full-length 64-bit	not shared
6	PCI Express hot-plug	3.3	20 Gbps	x4 lane width	not shared
7	PCI Express hot-plug	3.3	40 Gbps	x8 lane width	not shared

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ280 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration (1) (2) (3)	**Slot Priority**				
	1	**2**	**3**	**4**	**5**
	slot numbers				
RCIM only (3) (4)	2/3/4/5	n/a			
1 PCI-X slot filled (USB on, eth0 down, eth1 up)	2/3/4/5	n/a			
2 PCI-X slots filled (USB on)	2	4/5	n/a		
	3	2/4/5			
	4	5			
3 PCI-X slots filled (USB on)	2	4/5	3	n/a	
	4	5	2/3		
4 PCI-X slots filled (USB on)	4	5	3	2	n/a
Legacy slot 1 + 1 PCI-X slot filled (USB on)	2/3/4/5	1	n/a		
Legacy slot 1 + 2 PCI-X slots filled (USB on)	2	4/5	1	n/a	
	3	2/4/5	1		
	4	5	1		
Legacy slot 1 + 3 PCI-X slots filled (USB on)	2	4/5	1	3	n/a
	4	5	1	2/3	
Legacy slot 1 + 4 PCI-X slots filled (USB on)	4	5	1	3	2

(1) This table reflects only single-IRQ cards. If multi-IRQ cards are installed, toggle USB on/off in the BIOS to determine the best priority for these boards.

(2) Slot priorities are demoted if USB is disabled in the BIOS.

(3) Slot 1 is always assigned IRQ 177 and shares the vector with USB. Not recommended for RCIM.

(4) With only an RCIM configured, use USB on, SCSIB instead of SCSIA, NIC2 (eth1) instead of NIC1 (eth0). If using eth0 with USB enabled, slot 4 will have priority 2; with USB disabled, the RCIM will have priority 4 or worse.

iHawk Model HQ265

Specifications

Manufacturer/Model	**Dell™ PowerEdge™ 2650**
Processor	**Intel® Xeon™**
Total number of PCI slots	3
PCI/PCI-X	3
PCI Express	0
Hot-plugs	0
Number of PCI buses	2
Embedded devices	2 Broadcom Tigon3 Gigabit Ethernet controllers 1 Dual Channel U160 SCSI/RAID controller 1 Serverworks CSB5 IDE controller 1 ATI-Rage XL VGA controller 1 Serverworks OSB4/CSB5 USB hub with 4 controllers

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ265 system. The slots are numbered on the main board adjacent to each slot.

Slot Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length	Shares PCI bus with
1	PCI-X	3.3	133 MHz	full-length 64-bit	slot 2
2	PCI-X	3.3	133 MHz	full-length 64-bit	slot 1
3	PCI-X	3.3	133 MHz	full-length 64-bit	not shared

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ265 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving the highest priority.

Configuration	Slot Priority (1)		
	1	2	3
	slot numbers		
RCIM only	1/2/3	n/a	
1 slot filled (USB off)	1/2/3	n/a	
2 slots filled (USB off)	2/3	1	n/a
	2	3/1	
	3	1	
3 slots filled (USB off)	2	3	1

(1) PCI slots receive IRQ assignments that are lower in priority than the embedded devices, even when disabling the server management NICs and USB.

iHawk Model HQ067

Specifications

Manufacturer/Model	**Dell™ Precision™ 670**
Processor	**Intel® Xeon™**
Total number of PCI slots	6
PCI/PCI-X	4
PCI Express	2
Hot-plugs	0
Number of PCI buses	5
Embedded devices	1 Dell Serial ATA 150 controller 1 Adaptec U320 SCSI with RAID 0,1 1 Intel Pro 1000 Gigabit Ethernet controller 1 Analog Devices AC97 Audio controller 8 USB 2.0 ports Optional: U320 RAID 0,1,15,10 Sound Blaster Audigy 2D with 1394a port

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ067 system.

Slot Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length	Shares PCI bus with
1	PCI Express	3.3	80 Gbps	x16 lane width	not shared
2	PCI 2.2	5	33 MHz	half-length 32-bit	shared vector
3	PCI Express	3.3	20 Gbps	x4 lane width	not shared
4	PCI-X	3.3	100 MHz	full-length 64-bit	not shared
5	PCI-X	3.3	100 MHz	full-length 64-bit	not shared
6	PCI-X	3.3	100 MHz	full-length 64-bit	not shared

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ067 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration (1)	**Slot Priority**			
	1	**2**	**3**	**4**
	slot numbers			
RCIM only	5	n/a		
1 PCI-X slot filled (2)	5	n/a		
2 PCI-X slots filled	5	6	n/a	
	4	5/6		
3 PCI-X slots filled	5	6	4	n/a
Legacy slot 2 + 1 PCI-X slot filled	5	2	n/a	
Legacy slot 2 + 2 PCI-X slots filled	5	6	2	n/a
	4	5/6	2	
Legacy slot 2 + 3 PCI-X slots filled	5	6	2	4

(1) This scheme was determined with the PCI Express graphics card installed in slot 1. There is no embedded video controller.

(2) The embedded Ethernet always receives the highest IRQ priority level when only one card is installed on the bus.

iHawk Model HQ047

Specifications

Manufacturer/Model	**Dell™ Precision™ 470**
Processor	**Intel® Xeon™**
Total number of PCI slots	4
PCI/PCI-X	2
PCI Express	2
Hot-plugs	0
Number of PCI buses	3
Embedded devices	1 Dell Serial ATA 150 controller 1 Adaptec U320 SCSI with RAID 0,1 1 Intel Pro 1000 Gigabit Ethernet controller 1 Analog Devices AC97 Audio controller 1 Intel ICH5 IDE controller 8 USB 2.0 ports Optional: U320 RAID 0,1,15,10 Sound Blaster Audigy 2D with 1394a port

PCI Bus and Slot Layout

The table below provides details of the PCI slots and buses found on the iHawk HQ047 system.

Slot Number	PCI Type	Voltage	Maximum PCI Bus Speed	PCI Bus Length	Shares PCI bus with
1	PCI Express	3.3	80 Gbps	x16 lane width	not shared
2	PCI 2.2	5	33 MHz	half-length 32-bit	shared vector
3	PCI Express	3.3	20 Gbps	x4 lane width	not shared
4	PCI-X	3.3	100 MHz	full-length 64-bit	not shared

IRQ Priority Management

The table below shows a variety of configuration options on the iHawk HQ047 system and the corresponding IRQ priority levels that are assigned to each slot.

The device requiring the highest IRQ priority level (usually the RCIM) should be located in the slot receiving Priority 1.

Configuration (1) (2)	**Slot Priority**	
	1	**2**
	slot numbers	
RCIM only	4	n/a
1 PCI-X slot filled (3)	4	n/a
Legacy slot 2 + PCI-X slot filled	4	2

(1) This scheme was determined with the PCI Express graphics card installed in slot 1. There is no embedded video controller.

(2) When SATA, USB and ICH5 embedded devices are all enabled, devices like uhci_hcd, ehci_hcd, libata and ICH5 are sharing vectors. Slot 2 always shares a vector with the Intel ICH5 audio controller. The only way to eliminate the shared vector is to disable the audio controller.

(3) The embedded Ethernet always receives the highest IRQ priority level when only one card is installed on the bus.

3
Performance Factors in Configuring iHawks

3
Performance Factors in Configuring iHawks

Improper system administration and peripheral device configuration can have unexpected and adverse effects on real-time performance. Understanding the PCI subsystem, managing hardware interrupts and maximizing bandwidth on the PCI bus are key to making the PC architecture a reliable real-time platform.

Chapter 2 provides the PCI bus and slot layouts of individual iHawk systems and recommendations for configuring boards based on the way in which each system allocates IRQ priorities.

Explanations of the terms used in this discussion and in Chapter 2 are provided in the Glossary.

iHawk Overview

Concurrent's iHawk systems are symmetric multiprocessors (SMPs) ideally suited for real-time data acquisition, simulation and industrial systems applications. iHawks feature from one to eight Intel® Xeon™ or AMD Opteron™ processors in a single rackmount or tower enclosure with integral 32/64-bit PCI slots and PCI expansion options.

Each iHawk includes Concurrent's Real-Time Clock and Interrupt Module (RCIM). This unique multifunction card provides programmable interrupts and real-time clocks as well as synchronized timekeeping across systems. The RCIM mounts in a standard iHawk PCI slot.

At the heart of each iHawk system is Concurrent's RedHawk Linux or SUSE Linux Enterprise Real Time operating system that provides many features for maximizing determinism and real-time performance for mission-critical solutions. The power of the operating system is fully realized on a properly configured hardware platform.

How to Configure iHawks for Performance

There are several factors involved in configuring an iHawk system to achieve the best performance for real-time applications. For example, the slot in which a device is installed can substantially affect its performance in terms of bandwidth and hardware interrupt priority.

In Chapter 2, we have provided recommendations for configuring each individual iHawk system. In this section, we have listed some basic principals based on how the PCI subsystem operates that determine whether a device will run at full capacity and receive the priority handling it deserves. More complete explanations are provided later in this chapter.

In many cases, simply moving a board to a different slot or changing a single setting can increase the total bandwidth to that device or to the system as a whole. It can also make the difference between a bad interrupt priority scheme and a good one.

Sometimes a combination of changes is required. Trade-offs may need to be made. For example, you may have to settle for placing a card in a slot that receives a low hardware interrupt (IRQ) priority level in order to use a device that must be installed in the higher priority slot.

Complex systems are more difficult to optimize than simpler systems; however, experimenting with different combinations of slot populations will reveal a unique pattern of IRQ assignments. In some cases, installing a card that is not needed can force a better pattern of IRQ assignments. Once a pattern emerges, you can then swap cards in order to match a device to the priority it should receive. Note that a pattern of IRQ assignments may apply only when a certain number of boards is configured, and any change to that number may produce a different pattern.

Follow these basic recommendations when attempting to optimize system configuration:

Install devices in slots according to IRQ priorities, installing the RCIM in the slot receiving the highest priority.

The order in which IRQs are serviced is determined by the priorities that are assigned by the Linux kernel. Because of the deterministic event synchronization capabilities of the RCIM, it should receive the highest possible IRQ priority and occupy a slot that either does not share interrupts or shares with a device that can be deactivated.

Determine which devices on your system require the best response and adjust the system configuration accordingly. This may involve moving a board to a slot that receives a higher IRQ priority level or changing certain BIOS settings. For more information about how IRQs are prioritized, refer to the section “IRQ Priorities.” BIOS setting recommendations are given in the section “BIOS Setting Recommendations”.

While IRQ priorities are important, also consider bandwidth requirements of the device. For some devices such as sound cards, obtaining the best bandwidth is more important than a high hardware interrupt priority.

Manage IRQ sharing.

More than one device may be configured to use the same vector to process interrupts. This can cause jitter in real-time applications. Devices sharing an interrupt vector will have the same Linux IRQ number. Because these IRQ numbers are spinlocked, it may not be possible for a device to send an interrupt until the spinlock is released by another device sharing that vector.

Use **`lsirq(8)`** to determine if vectors are shared and, if so, move boards or disable unused embedded devices that are sharing that vector if possible. Refer to the section “Shared Vectors and lsirq(8)” for details.

Populate slots that have their own PCI bus first.

When a bus is shared between two PCI slots, the devices in these slots compete with each other for bus cycles. Therefore, not having to share a bus means better performance for the device.

If a bus must be shared, match device mode, speed and activity level.

You can install cards of different operating speeds on the same bus; however, the bus will operate at the speed of the slowest card on that bus. For example, if one card on the bus has an operating speed of 66 MHz and the other card has an operating speed of 100 MHz, the bus can operate only at 66 MHz. Also, if a PCI card is installed on the same bus with a PCI-X card, the bus runs in PCI mode. For more information, refer to the section “Shared Data Paths on PCI and PCI-X Parallel Buses.”

Also, pairing a busy device with a less busy device reduces bus contention by not having both devices on a single bus continuously performing numerous reads and writes.

Avoid installing two devices requiring high PCI latencies on the same bus.

The length of time devices control the PCI bus can affect throughput. How long a device must wait to use the bus needs to be weighed against how much data can be moved in a single transfer. This BIOS setting can be modified if adjustments are needed; however, it is generally not recommended. For more information, refer to the section “Latency.”

Disable unused embedded devices.

Some slots tend to receive IRQs lower in priority to many embedded devices or may share an interrupt vector with an embedded device. Embedded devices that are not in use should normally be disabled in the BIOS.

Minimize hardware interrupt processing.

The lowest priority hardware interrupt will preempt the highest priority process (see “IRQ Priorities”). Operating peripheral devices that are not needed by a real-time application will result in lost CPU cycles and a decrease in determinism. High speed devices such as hard drives and network controllers can send a stream of interrupts that can cause important real-time interrupts to be lost or delayed in delivery. Even a device without a loaded driver can send interrupts that must be ACKed and ignored; for example, a downed ethernet device attached to an active network.

Follow recommended BIOS “rule of thumb” settings.

In a finely tuned real-time operating system, the settings in the BIOS can affect the performance of your applications. Although there is no BIOS standard for all iHawk systems, the guidelines given in the section “BIOS Setting Recommendations” have shown to be the most effective for the real-time features in Concurrent’s Linux operating systems.

PCI, PCIe and PCI-X

The following sections discuss the basic operations of these three peripheral bus architectures and the impact they have on real-time applications.

The following technologies are used in iHawk systems to connect peripheral devices to the system bus.

PCI	Peripheral Component Interconnect
AGP	Accelerated Graphics Port (modified PCI direct connection)
PCI-X	Peripheral Component Interconnect Extended
PCI-X 2.0	PCI-X version 2 (being replaced by PCIe)
PCIe	PCI Express
PCIe 2.0	PCI Express version 2 (doubles the bandwidth of PCIe)

Chapter 2 provides the slot types used by each of the iHawk models.

PCI Bus Architecture

A bus is a data path shared by more than one device. Only one device may have access to the bus at any given time, so devices must contend for the bus. The total number of devices that can be attached to the bus is limited and does not scale well. PCI is a parallel bus architecture.

The PCIe architecture is commonly referred to as a “bus” but is not a true bus. PCIe is a switched point-to-point connection. It connects only two devices and no other device can share this connection.

On a motherboard using standard PCI slots, all PCI slots are connected to the PCI bus and share the same data path. On a motherboard with PCIe slots, each PCIe slot is connected to the motherboard chipset using a dedicated lane, not sharing this lane (data path) with other PCIe slots.

Shared Data Paths on PCI and PCI-X Parallel Buses

The PCI system provides the capability for devices to share interrupts. This necessity originated from the early uniprocessor systems, which used the 8259 interrupt controller. The 8259 had only 16 pins available for use by external interrupt sources and many of those were already dedicated to devices permanently embedded in the i386 architecture. In order to add devices, they must be able to share interrupts, and a key element of the PCI subsystem design is its own internal interrupt system.

A PCI bus has four wires connected to four pins (labeled A, B, C, D) on a PCI adapter card. That means that four cards can be installed on the same bus and each one can have its own wire for sending and receiving hardware interrupts. If a fifth card is added to a bus, at least two of the devices must share a wire, that is, share hardware interrupts. Refer to the section “Shared Vectors and lsirq(8)” for more information about how devices share hardware interrupts.

PCI and PCI-X both utilize the shared bus technology and provide a “plug and play” capability. PCI and 32-bit PCI-X slots are physically the same and can be used interchangeably; however, whenever PCI slots share the same bus, the bus can only operate at the speed and capabilities of the slowest device. The 64-bit PCI-X slots are physically longer than the PCI slots and cannot be used interchangeably with PCI slots.

PCI bus speed and mode should be taken into consideration when configuring your system. Boot messages such as the following:

```
Warning: PCI-X is running in PCI mode which impacts performance.
```

indicate that changes should be made. As an example, the RCIM operates in PCI mode. If the RCIM is installed in a PCI-X slot that shares its bus with a high speed device, the high speed device will operate in the slower PCI mode. Ideally, the RCIM should be placed in a slot with its own PCI bus or with the least possible contention with other devices. The RCIM does not need to hold the PCI bus for long periods and its presence is fairly benign; the RCIM primarily affects PCI bus speed and mode.

Unique Data Paths on PCIe

PCIe makes it much faster and easier for data to get around the system by combining scalable, high-bandwidth data paths, packetized data protocols, and compatibility with PCI hardware and device drivers.

PCIe is software compatible with PCI, but not plug compatible. Rather than the shared, parallel bus structure of PCI, PCIe provides a high-speed switched architecture. Each PCIe link is a serial communications channel made up of two differential wire pairs that provide 2.5 Gbits/sec in each direction. Up to 32 of these “lanes” may be combined in x2, x4, x8, x16 and x32 configurations, creating a parallel interface of independently controlled serial links. The bandwidth of the switch backplane determines the total capacity of a PCIe implementation.

PCIe sends all control messages, including interrupts, over the same links used for data. The serial protocol can never be blocked, so latency is still comparable to PCI, which has dedicated interrupt lines.

Bandwidth

The speed gap between CPU cores and system memory has always been a design contraint. Cacheing improves performance, but cache coherence protocols complicate the design of multiprocessor systems. Also, traditional multiple shared bus architectures do not scale well above 16 cores. The Intel Xeon uses such a bus architecture.

The AMD Opteron architecture employs hypertransport link technology, which is similar to PCIe in that it is not a bus, but a switched point-to-point link. Quad-core Opteron systems were the first to achieve 32 cores using the K8 NUMA (Non Uniform Memory Architecture) and a unique cache coherence protocol.

In recent years, new memory technology has allowed a significant increase in system bus speeds. This has greatly decreased memory access latencies and increased memory bandwidth and system throughput. However, PCI speed has lagged far behind.

The PCI shared parallel bus architecture suffers from a fixed maximum bandwidth that data endpoints in a system must share (see Table 3-1). As you add boards to a PCI bus you reduce the bandwidth available to each PCI function. You cannot simply increase the PCI bus frequency to match increases in the system bus frequency, because slower PCI devices cannot simply speed up. In fact, the maximum speed of any PCI bus is limited by the maximum operating speed of the slowest PCI device on the bus.

PCIe sends all control messages, including interrupts, over the same links used for data. The serial protocol can never be blocked, so latency is still comparable to PCI, which has dedicated interrupt lines.

Table 3-1 PCI, PCI-X, AGP and PCIe Maximum Bandwidths

Architecture	Maximum Bandwidth
PCI 32 33MHz	133 MB/s
PCI 32 66MHz	266 MB/s
PCI 64 33MHz	266 MB/s
PCI 64 66MHz	533 MB/s
PCI-X 64 66MHz	533 MB/s
PCI-X 64 100MHz	800 MB/s
PCI-X 64 133MHz	1066 MB/s
AGP 1X	266 MB/s
AGP 2X	533 MB/s
AGP 4X	1066 MB/s
AGP 8X	2133 MB/s
PCIe x1	250 [500]* MB/s
PCIe x2	500 [1000]* MB/s
PCIe x4	1000 [2000]* MB/s
PCIe x8	2000 [4000]* MB/s
PCIe x16	4000 [8000]* MB/s

* NOTE – Since PCIe is a serial based technology, data can be sent over the bus in two directions at once. The first number is the bandwidth in one direction; the second number in brackets is the combined bandwidth in both directions. PCIe bandwidth is not shared the same way as in PCI, so there is less congestion on the bus.

Determinism

In addition to bandwidth limitations, real-time problems occur with shared-bus systems. External events happen at random times, and a bus system can respond to only one event at a time. If a data transfer is in progress and a higher priority transfer needs the bus, circuitry or software must suspend the lower priority data until the high priority finishes and then retransmits the blocked data. This contention for the bus is proportional to the number of nodes in the system. The latency of suspended data may become intolerable in some real-time situations.

PCIe eliminates these problems by providing a separate interconnect path for each point-to-point data-transfer possibility so that multiple data transfers can occur simultaneously.

Latency

Latency refers to how long a PCI device can hold the PCI bus before handing it over to another device. The longer the latency, the longer the device can retain control of the bus.

Normally, the PCI latency timer is set to 32 cycles. This means the active PCI device has to complete its transactions within 32 clock cycles or hand it over to the next PCI device.

In general, a longer latency yields better performance, but the optimal value for every system is different. In some cases, a long latency can reduce performance as the other PCI devices queueing up may be stalled for too long. This is especially true with systems having many PCI devices or PCI devices that continuously write short bursts of data to the bus. Such systems would work better with shorter PCI latencies as they allow rapid access to the bus.

Time-critical PCI devices usually require priority access to the PCI bus, which may not be possible if the bus is held up by another device for a long period. The default of 32 cycles is usually best for these cases.

You can experiment with different settings, benchmarking the devices' performance after each change to determine the optimal PCI latency time for your system.

PCI latency is usually configurable in the BIOS or by using the **`setpci(1)`** command after boot.

PCI Expansion

PCI expansion units add more slots to a system. These additional slots will appear to the system as a normal PCI slot.

Understanding IRQ Management

The speed in which a hardware interrupt is serviced is crucial to performance. This section describes IRQs and how they are processed to help you understand how modifications to the system configuration can make a difference.

Chapter 2 contains recommendations for optimizing configuration on each iHawk system based on its IRQ management scheme.

Explanations of many of the terms used in this discussion and in Chapter 2 are provided in the Glossary.

IRQ Basics

An IRQ is an interrupt request. It is used in many contexts and can be the source of some confusion.

At the bus level, IRQ refers to an "interrupt request line," which is a wire connecting a peripheral device to a hardware interrupt controller. This is a number between 0-15 and is assigned by the BIOS.

At the processor level, IRQ refers to an "interrupt vector," which is an integer from 0-255 stored in a register on the interrupt controller. This interrupt vector points to the interrupt

service routine(s) (ISRs) used for that interrupt, and also determines the priority of the interrupt. A vector can be shared by more than one interrupt; this is called a “shared vector.”

At the Linux operating system level, the term “Linux IRQ number” refers to the IRQ number that is mapped to the actual interrupt vector (APIC pin) stored in the interrupt controller. By looking at Linux IRQ numbers, we can determine how IRQs are prioritized in the system and whether vectors are shared.

These areas are discussed at length in the following sections.

Message Signaled Interrupts (MSI/MSI-X)

All drivers written for PCIe use Message Signaled Interrupts (MSIs), which involves a point-to-point switching connection instead of a shared bus technology. An MSI-capable hardware device sends an inbound Memory Write on its PCI bus instead of asserting an IRQ signal on a device IRQ pin. Because MSI is generated in the form of a Memory Write, all transaction conditions, such as a Retry, Master-Abort, Target-Abort or normal completion, are supported.

MSIs do not share linux IRQ numbers and are therefore not subject to the IRQ spinlock contention and non-deterministic interrupt response times.

MSI-X is an extension to MSI.

Shared/Nonshared IRQs

MSIs eradicate shared IRQs, which is desirable for deterministic real time performance.

Both PCI/PCI-X and MSI/MSI-X devices can use MSIs, but the system must also support MSIs.

If the system does not support MSI, a PCI/PCI-X device that supports MSIs must also support the pin IRQ assertion interrupt mechanism (IRQ lines connected to IOAPIC pins) for backward compatibility.

Depending upon the device and its driver, BIOS IRQ line numbers may or may not be utilized. A ‘cat’ of **`/proc/interrupts`** on a system with MSI devices shows the Linux IRQ number the device uses and the MSI interrupt mode (MSI or MSI-X). For example:

```
# cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
...
185:          0          0          0          0     IO-APIC-level  uhci-hcd
193:         51          0          0          0            PCI MSI  aic79xx
...
```

PCIe devices do not have IRQ lines, so PCIe endpoints must use INTx emulation (in-band messages) instead of IRQ pin assertion for backward compatibility.

Using INTx emulation requires interrupt sharing among devices connected to the same node (PCI bridge) while MSI is unique (non-shared) and does not require BIOS configuration support. So PCIe devices using INTx emulation can share IRQs whereas PCIe devices using MSI will have unique (non-shared) IRQs.

PCI/PCI-X devices using MSIs will also receive unique (non-shared) IRQs, even though the BIOS will associate an IRQ line with the device.

Using MSI enables the device functions to support two or more vectors, which can be configured to target different CPUs to increase scalability.

MSI can be overridden at boot time by issuing the **`pci=nomsi`** kernel boot parameter on the grub line. All PCIe devices will then revert to INTx emulation and all PCI devices will revert to pin IRQ assertion.

System Requirements for MSI/MSI-X Support

MSI/MSI-X requires support from both system hardware and individual hardware device functions as well as the kernel. All iHawk systems support MSIs. All RedHawk 4.1 and later pre-built kernels include this support.

For the case where a function implements both MSI and MSI-X capabilities, the PCI subsystem enables a device to run either in MSI mode or MSI-X mode but not both. A device driver determines whether it wants MSI or MSI-X enabled on its hardware device. Once a device driver requests MSI, for example, it is prohibited from requesting MSI-X; so a device driver will not ping-pong between MSI mode and MSI-X mode during a run-time.

MSI/MSI-X support in the kernel requires that the PCI_MSI kernel parameter be enabled. Since the target of an MSI address is the local APIC CPU, enabling MSI/MSI-X support in the Linux kernel is dependent on whether existing system hardware supports local APIC. To verify that your system supports local APIC operation, test that it runs when X86_LOCAL_APIC is enabled in the kernel. In an SMP environment, X86_LOCAL_APIC is automatically set; in a UP environment, this must be set manually.

With X86_LOCAL_APIC set, setting PCI_MSI enables the VECTOR based scheme and the option for MSI-capable device drivers to selectively enable MSI/MSI-X.

Note that the setting of X86_IO_APIC is irrelevant because the MSI/MSI-X vector is allocated during runtime and MSI/MSI-X support does not depend on BIOS support. This key independency enables MSI/MSI-X support on future IOxAPIC free platforms.

PCI Capability List and MSI/MSI-X Capability Structure

An MSI capable device function indicates MSI support by implementing the MSI/MSI-X capability structure in its PCI capability list.

In systems which support MSI, the bus driver is responsible for initializing the message address and message data of the device function's MSI/MSI-X capability structure during device initial configuration.

The device function may implement both the MSI capability structure and the MSI-X capability structure; however, the bus driver should not enable both.

The MSI capability structure contains a Message Control register, Message Address register and Message Data register. These registers provide the bus driver control over MSI. The Message Control register indicates the MSI capability supported by the device. The Message Address register specifies the target address and the Message Data register specifies the characteristics of the message. To request service, the device function writes the content of the Message Data register to the target address. The device and its software driver are prohibited from writing to these registers.

The MSI-X capability structure is an optional extension to MSI. It uses an independent and separate capability structure. Key advantages to implementing the MSI-X capability structure over the MSI capability structure include the following:

- Support for a larger maximum number of vectors per function.
- The ability for system software to configure each vector with an independent message address and message data, specified by a table that resides in Memory Space.
- Per-vector masking is an optional extension of MSI but a required feature for MSI-X. Per-vector masking provides the kernel the ability to mask/unmask a single MSI while running its interrupt service routine. If per-vector masking is not supported, the device driver should provide the hardware/software synchronization to ensure that the device generates MSI when the driver wants it to do so.

MSI/MSI-X Mode vs. Legacy Mode

Figure 3-1 shows the events that switch the interrupt mode on the MSI-capable device function between MSI mode and PIN-IRQ assertion mode.

A device operates by default in legacy mode. Legacy in this context means PCI pin-irq assertion or PCIe INTx emulation. A successful MSI request (using pci_enable_msi()) switches a device's interrupt mode to MSI mode. A pre-assigned IOAPIC vector stored in dev->irq will be saved by the PCI subsystem and a new assigned MSI vector will replace dev->irq.

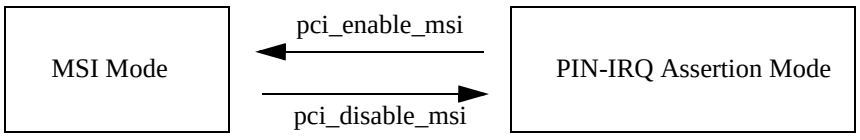


Figure 3-1 MSI Mode vs. Legacy Mode

Figure 3-2 shows the events that switch the interrupt mode on the MSI-X capable device function between MSI-X mode and PIN-IRQ assertion mode (legacy).

A device operates by default in legacy mode. A successful MSI-X request (using pci_enable_msix()) switches a device's interrupt mode to MSI-X mode. A pre-assigned IOAPIC vector stored in dev->irq will be saved by the PCI subsystem; however, unlike MSI mode, the PCI subsystem will not replace dev->irq with assigned MSI-X vector because the PCI subsystem already writes the 1:1 vector-to-entry mapping into the field 'vector' of each element specified in second argument.

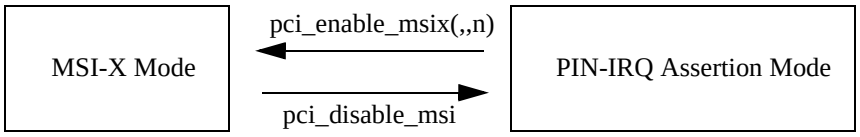


Figure 3-2 MSI-X Mode vs. Legacy Mode

How IRQs are Assigned

IRQ assignments hinge on logical slot addresses read by the kernel from the BIOS during boot. These logical slot assignments might change for each device whenever a device is added or removed from a system. Once a hardware configuration is established, the logical slot assignments will be the same from boot to boot.

The behavior of the logical slot assignments varies from system to system, BIOS to BIOS, and kernel to kernel. It is the interaction of the hardware, the BIOS, and the kernel boot code that produces the resulting IRQ assignment.

The BIOS uses an EEPROM called the ECSD to store information on each device detected by the BIOS. The ECSD is reprogrammed whenever a device is added or removed, and a new set of resources is allocated for the new configuration. It is the information stored in the ECSD that determines the logical slot assignment for a device.

The ECSD must be reprogrammed whenever an embedded device is enabled or disabled in the BIOS, and whenever PCI cards are added, moved or removed from the PCI bus. The ECSD is read or reprogrammed early in the POST (Power On Self Test).

The ECSD can also be "cleared" in some BIOSes, which will also result in the BIOS reprogramming it on the next boot. Sometimes clearing the ECSD will eliminate resource allocation conflicts. Unless some strange error occurs, the ECSD will come up with the same set of resource allocations for any given configuration. Sometimes, the only way to clear up a PCI resource conflict is to move cards to different slots.

When the kernel begins executing its bootstrap code, it reads a logical slot presented by the BIOS and assigns this slot a GSI (Global System Interrupt) number. This GSI number is part of the ACPI specification (see the next section) and can be thought of as a "plug and play number." The GSI number becomes a label for one of the IOAPIC interrupt pins. An interrupt vector is chosen to be associated with this interrupt pin and is programmed into the register wired to that pin. The vector is then mapped to a Linux IRQ number.

The mapping from the kernel to the actual interrupt vector looks like this:

Linux IRQ number (0-255) -> GSI number (or IOAPIC pin) -> vector

From the standpoint of the bus, the mapping looks like this:

Device interrupt pin -> Bus IRQ number (0-15) -> GSI number (or IOAPIC pin) -> vector

The simple view of the mapping from the view of the kernel and the BIOS is:

Bus IRQ number (0-15) <-> vector <-> Linux IRQ number (0-255)

The bus IRQ number (the number the BIOS recognizes) is merely a label for the wire used to connect the device to the APIC. It has nothing to do with interrupt priority unless the system is booted in PIC mode. When the system is booted in PIC mode, the Linux IRQ number is the bus IRQ number. The priority is still vector based, so the 8259 interrupt priority scheme used in earlier systems is no longer applied, unless the system has no IOAPIC and is using the 8259 or its equivalent.

ACPI

ACPI (Advanced Configuration and Power Interface) is a standard that provides two main functions:

- hardware discovery and resource allocation (Plug and Play)
- power management

It is ACPI that sets up the tables used by the operating system and device drivers to send and receive interrupt requests. It programs IOAPICS with the legacy ISA and EISA IRQs, the system timer, and the level triggered interrupts used by the PCI subsystem. ACPI supercedes APM (Auxillary Power Management).

ACPI is the glue that binds the operating system to the BIOS. As such, a basic understanding of ACPI can be very helpful in fully understanding IRQs in Linux. For more information on ACPI, visit the following links:

- The offical ACPI specification: http://acpi.info/
- ACPI Project at SourceForge (primarily deals with power management): http://sourceforge.net/projects/acpi/
- Reprint from the 2004 Ottawa Linux Symposium: *The State of ACPI in the Linux Kernel*: http://www.linuxsymposium.org/proceedings/reprints/Reprint-Brown-OLS2004.pdf

For recommendations for ACPI-related BIOS settings, see the section “BIOS Setting Recommendations”.

IRQ Priorities

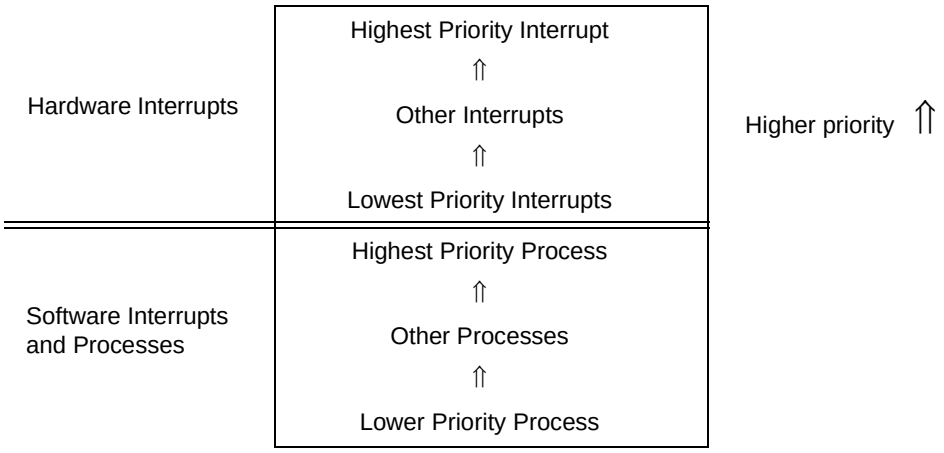
IRQs are prioritized at the hardware level. There are 16 priority classes numbered from 0-f, with each class containing 16 interrupt vectors. The priority class is determined by taking the first 4 bits of the vector (the vector “d8” belongs to priority class “d”). These vector and class designations are internal and not viewable by the user.

In the Intel P4 architecture, the interrupt vectors contained in each priority class are also prioritized. The priority within the class is determined by left shifting the vector 4 bits; so, vector “d9” belongs to class “d” and is higher in priority than vector “d8”. The vector “ff” has the highest possible interrupt priority, and “00”, the lowest. There are 256 possible interrupt vectors and 256 different priority levels.

Hardware interrupts can be either “maskable” or “non-maskable.” Maskable hardware interrupts are those associated with peripheral devices. Non-maskable hardware interrupts (NMIs) are associated with low level system services such as the system timer, or a memory refresh, and will preempt maskable interrupts. The “System Management Interrupt,” or SMI, is another class of NMI. SMIs are completely outside of the software architecture, and control such things as board level power management. SMIs preempt all hardware interrupts.

Because hardware interrupts are higher priority than software interrupts, any hardware interrupt will preempt any software interrupt or process. A hardware interrupt can only be preempted by a higher priority hardware interrupt.

Priority inversion of real-time processes can come about because the differing hardware and software priority schemes on a system can interact in unexpected ways. The figure below illustrates the split priority scheme in this type of architecture.



As the figure shows, a high priority process can be preempted by a low priority process’ IRQs.

Worse yet, a low priority process could be using some peripheral device that might have the highest priority IRQ in the system. This means that the low priority process IRQs will always preempt the higher priority process IRQs.

For a complete discussion of how software interrupts are processed and how these interrupts can be routed away from a specific CPU through CPU shielding, refer to Chapter 2 of the appropriate Concurrent operating system User's Guide.

Shared Vectors and lsirq(8)

A shared wire on a PCI bus is called an "IRQ channel." Two devices can have the same BIOS assigned IRQ line number and not be on the same IRQ channel. Under the uniprocessor 8259 system, that was not true; if two devices shared an interrupt pin, they had the same IRQ line number. With the advent of the APIC in SMP systems, more pins were available for external interrupt sources. Many PCI systems have a dedicated APIC for critical I/O subsystems and many PCI buses have only one slot having its own APIC interrupt pin. In these cases, interrupts are not shared.

APICs do not use the IRQ line number to trigger the correct interrupt service routine (ISR). Instead, a "vector" is stored in a register on the APIC that is associated with only one of the pins on that chip. The vector, in combination with the device PCI ID number is used instead of the IRQ line number to determine the correct ISR. A shared vector occurs whenever there is more than one interrupt source assigned to use a given vector.

IRQs are spinlocked, so if a second device wants to use its IRQ, it must wait until the first device using the same IRQ releases the spinlock. Because of spinlock contention, shared vectors increases IRQ latency and non-deterministic behavior in program execution and are not desirable for real time devices.

A shared vector condition is easily recognized when two or more devices are assigned to the same Linux IRQ number. The devices can be those installed in PCI/PCI-X slots and embedded devices.

Linux IRQ numbers can be found in boot messages and in the leftmost column of output from **`/proc/interrupts`**:

```
# cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
  0:         51          0    2448698          0    IO-APIC-edge  timer
  3:          0          0          0          0    IO-APIC-edge  KGDB-stub
  4:        284          0          0          0    IO-APIC-edge  serial
  9:          0          0          0          0   IO-APIC-level  acpi
 14:          0          0         21          1    IO-APIC-edge  ide0
 50:      16943          0          0          0   IO-APIC-level  eth0
177:          0          0          0          0   IO-APIC-level  uhci_hcd
185:          0          0          0          0   IO-APIC-level  uhci_hcd
193:          0          0          0          0   IO-APIC-level  uhci_hcd
201:          0          0          0         16   IO-APIC-level  ehci_hcd
209:         29          0          0          0   IO-APIC-level  ioc0
217:          0       3861          0          0   IO-APIC-level  ioc1
225:          0          0          0          0   IO-APIC-level  btp
233:          0          0          0          0   IO-APIC-level  rcim

NMI:        285        265        194        107   Non-maskable interrupts
LOC:    2448418    2448383    2448431    2448353   Local interrupts
RES:         93         77         81        106   Rescheduling interrupts
CAL:        112        113        116         58   function call interrupts
TLB:         95        151         97         99   TLB shootdowns
TRM:          0          0          0          0   Thermal event interrupts
SPU:          0          0          0          0   Spurious interrupts
```

```
ERR:          0          0          0          0   Error interrupts
MIS:          0          0          0          0   APIC errata fixups
```

The **/proc/interrupts** display, however, only shows devices that are registered; that is, currently loaded in the system. If a device driver is not loaded at the time you are viewing **/proc/interrupts**, you will not see that it may share interrupts when it is loaded. A more complete and convenient method of viewing Linux IRQ numbers is by using the **lsirq(8)** command.

Specifying **lsirq** with the **--map**/**-m** option produces a display similar to the following. The leftmost column is the Linux IRQ number, followed by a colon and the BIOS assigned IRQ number. *Bus*:*slot*.*function* and vendor/device names are also provided.

```
# lsirq -m
50:11    0b:07.0 Intel Corp. 82541GI/PI Gigabit Ethernet Controller (rev 05)
177:11   00:1d.0 Intel Corp. 82801EB/ER (ICH5/ICH5R) USB UHCI #1 (rev 02)
177:11   10:01.0 D-Link System Inc RTL8139 Ethernet (rev 10)
185:10   00:1d.1 Intel Corp. 82801EB/ER (ICH5/ICH5R) USB UHCI #2 (rev 02)
193:5    00:1d.2 Intel Corp. 82801EB/ER (ICH5/ICH5R) USB UHCI #3 (rev 02)
193:5    10:0d.0 ATI Technologies Inc Radeon RV100 QY [Radeon 7000/VE]
201:3    00:1d.7 Intel Corp. 82801EB/ER (ICH5/ICH5R) USB2 EHCI Controller (rev 02)
209:5    02:05.0 LSI Logic/Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 ...
217:11   02:05.1 LSI Logic/Symbios Logic 53c1030 PCI-X Fusion-MPT Dual Ultra320 ...
225:11   03:0b.0 SBS Technologies VME Bridge Model 618 (rev 44)
233:5    05:04.0 Concurrent Computer Corp RCIM II Realtime Clock & Interrupts Module
(rev 01)
```

The higher the Linux IRQ number, the higher the priority of the vector it references (i.e., IRQ 233 has a higher priority than IRQ 225).

In this example, the RCIM receives the highest IRQ priority in the system and is not shared with any other devices. Because of its deterministic capabilities, this is the most desirable position for the RCIM.

Using **lsirq** is the only reliable method of determining if devices share vectors. As you can see from this example, devices share Linux IRQ numbers 177 and 193. Devices having the same Linux IRQ number will also have the same BIOS IRQ line number; however, devices having the same BIOS IRQ line numbers may have different Linux IRQ numbers.

To display information about only the shared vectors, use the **--shared**/**-S** option, for example:

```
# lsirq -Sm
177:11   00:1d.0 Intel Corp. 82801EB/ER (ICH5/ICH5R) USB UHCI #1 (rev 02)
177:11   10:01.0 D-Link System Inc RTL8139 Ethernet (rev 10)

193:5    00:1d.2 Intel Corp. 82801EB/ER (ICH5/ICH5R) USB UHCI #3 (rev 02)
193:5    10:0d.0 ATI Technologies Inc Radeon RV100 QY [Radeon 7000/VE]
```

lsirq is useful for displaying the system configuration in a variety of formats and groupings. The **--irq**/**-i** option is particularly helpful because it displays device interrupt priorities on each bus. Other options allow you to focus on a class of devices (**--class**/**-c**) or groups of the same device (**--device**/**-d**). This can be helpful when configuring multiple display adapters or network interface cards. A complete list of **lsirq** options is provided at the end of this section and in the man page.

By examining which devices share IRQs and analyzing their requirements, you may find that moving boards to different slots or deactivating embedded devices that are not being

used results in better performance. After reconfiguring and rebooting the system, use **lsirq** to view the new IRQ assignments and determine if further changes are needed.

Note that it is possible to deactivate an embedded device, for example a network card, and install another device of that type in a PCI slot. This may result in the device not sharing an IRQ, or in having all devices on the system receive IRQ priorities more appropriate to their functionality.

It is easier to manage IRQ priorities for single IRQ boards than multi-IRQ boards. When two or more devices share a board, the way in which the IRQs on the board are assigned are system-dependent. You will need to experiment with different configurations to determine the best location for these boards.

The following options to **lsirq** produce output in various formats. Refer to the **lsirq(8)** man page for details:

-p, **--priority** list devices by ascending priority order (default display)

-S, **--shared** list IRQs grouped by shared vectors

-b [*bus* ...], **--bus** [*bus* ...]
list BIOS IRQ line numbers grouped by bus, or list only specified bus(es)

-i [*irq* ...], **--irq** [*irq* ...]
list IRQs grouped by bus in priority order, or list only specified IRQ(s).

-c [*class* ...], **--class** [*class* ...]
list PCI slots grouped by class, or list only specified class(es)

-d [*device* ...], **--device** [*device* ...]
list slots grouped by device, or list only specified device(s) in the form *class*:*vendor*:*device*

-s [*slot* ...], **--slot** [*slot* ...]
list logical slots in ascending order, or list only specified slot(s) in the form *bus*:*slot*.*function*

-R, **--rcim** list only RCIM details

-h, **--help** display usage information

-v, **--version** display the program version number

The following options can be used to modify the default output:

-r, **--reverse** reverse the order of display

-l, **--long** also show device IRQ priority and shared status

-m, **--map** show Linux IRQ to BIOS IRQ mappings

-n, **--numeric** show numeric IDs in the form *class*:*vendor*:*device*

Managing IRQ Assignments

IRQ priority assignments can be managed and shared vector assignments minimized through system configuration and administration.

Fortunately, the process of configuring a system to assign a high priority IRQ to a device usually also guarantees that the device will not share that IRQ (vector) with another device.

All devices in the system should be considered with respect to the effects they might have on the execution of a real-time application. You should try to configure hardware so that devices associated with high priority processes be of equally high IRQ priority. It may not be possible to control more than a few IRQ assignments through system configuration, but in most cases, several devices can be configured to have the most appropriate IRQ assignments.

Chapter 2 provides recommendations for configuring individual iHawk systems based on the way in which each system allocates IRQ priorities.

BIOS Setting Recommendations

You may find that certain systems require differing BIOS settings depending on the interaction of the kernel, installed devices and the BIOS.

In general, the following guidelines are designed to achieve maximum performance. The names of the BIOS settings differ between systems, so refer to your iHawk system documentation to determine which BIOS setting is involved.

- In general, power management settings should be left off (i.e., do not enable “Power NOW” or other such power management capabilities.
- ACPI sleep states should be limited to S1. When there is an “auto” setting, it would be best to ensure that S3 is never entered by explicitly setting “S1”.
- Systems should be enabled to be ACPI 2.0 compliant.
- The ACPI should be used for IRQ routing.
- ACPI supercedes the Intel MP (multiprocessor spec). If a choice is given for MP support, set it to 1.4.
- IOMMUs (sometimes referred to as “memory hole”) should be enabled on systems with greater than 4 GB of RAM. Turning this feature on with systems having 4 GB of RAM or less is not necessary, but problems with the PCI bus may occur if it is not set.
- Always select the most appropriate “installed OS” choice. If “linux” or its equivalent is not a choice, use the default “other”.
- Select large disk access mode as “other”, not DOS.
- For iHawks with Opteron processors, NUMA (non-interleaved memory nodes) is available. This means that each processor has direct access to a local physical memory array. On these systems, specify “non-interleaved” memory nodes when an “auto” choice is given. The desired functionality is “bank-interleaving”, not “node-interleaving”.
- Enable “logical processor”, “hyperthreading” or “dual-core” processor support.

Glossary

8259

The first interrupt controllers to appear in the PC architecture, used in uniprocessor x86 systems.

ACPI

ACPI (Advanced Configuration and Power Interface). A standard that provides hardware discovery and resource allocation (Plug and Play) and power management.

APIC

Advanced Programmable Interrupt Controller – a device used to generate and receive interrupt requests. Replaced the 8259 with the advent of multiprocessor (SMP) systems providing more available interrupt lines.

bandwidth

The amount of data that can be transmitted in a fixed amount of time.

BIOS

Basic Input/Output System – an interface between the operating system and the system hardware. The BIOS is key to setting up and managing IRQs and I/O devices. The BIOS setup utilities can only be reached during the Power On Self Test (POST), before the operating system boots.

device priority

The hardware priority used in the system for all maskable hardware interrupts: physical slots and embedded devices.

GSI

Global System Interrupt – another name for IRQ in the context of an APIC based interrupt delivery system.

GSI number

Another name for an IRQ line in the context of an APIC based interrupt delivery system. The GSI number is an integer assigned to name IRQ lines attached to an IOAPIC. The BIOS is not aware of GSI numbers; a legacy IRQ number is mapped from the BIOS to the GSI number.

hot-plug slot

When properly configured, a hot-plug slot allows the removal, replacement or addition of PCI or PCI-X expansion boards without powering down the system.

integrated mirroring

Provides simultaneous physical mirroring of two drives. Integrated mirroring functionality is provided by the system's hardware.

interrupt

An action generated by hardware or software that causes the CPU to stop what it's doing in order to perform work on behalf of whatever generated the interrupt.

interrupt controller

A device used to process interrupts.

IOAPIC

An APIC that serves as an interface between hardware I/O devices and any LAPIC. This device prioritizes, queues, and distributes interrupts in an equitable manner across all the LAPICs in the system. Allows any LAPIC to interrupt any other LAPIC (cross-processor interrupt).

IRQ

Interrupt request – a signal that data is about to be sent to or received by a peripheral device (or some action should be taken by the CPU to service the device). Each peripheral connection must be assigned an IRQ number. Two devices can share the same IRQ assignment, but both devices cannot operate simultaneously.

IRQ line

A wire connecting a peripheral device to a hardware interrupt controller. This is a number between 0-15 and is assigned by the BIOS.

ISR

Interrupt Service Routine – the actions taken in software or firmware as the result of an interrupt.

LAPIC

Local APIC – an APIC built into the CPU for the use of that CPU only. An LAPIC also has a timer function (local timer interrupt).

legacy device

Any device that uses edge triggered interrupts. Typically IDE, serial ports and system clocks. PCI uses level triggered interrupt or message signaled interrupts.

Linux IRQ number

An integer from 0-255 that is used as a handle for an interrupt vector.

message signaled interrupt (MSI/MSI-X)

Two separate mechanisms that enable a PCI Express device to request service by writing a system-specified DWORD of data to a system-specified address using a Memory Write Request. Compared to MSI, MSI-X supports a larger maximum number of vectors and independent message address and data for each vector.

NIC

Network Interface Controller – a device that is installed or integrated in a system to allow connection to a network.

PCI

Peripheral Component Interconnect – a peripheral bus that provides a high-speed data path between the processor and peripheral devices like video cards, sound cards, network interface cards and modems. PCI provides "plug and play" capability, runs at 33 MHz or 66 MHz and supports 32-bit and 64-bit data paths.

PCI Express

A high-speed peripheral interconnect from Intel that is software compatible with PCI, but not plug compatible. Rather than the shared, parallel bus structure of PCI, PCI Express provides a high-speed, switched architecture. Each PCI Express link is a serial communications channel made up of two differential wire pairs. Up to 32 of these "lanes" may be combined in x2, x4, x8, x16 and x32 configurations, creating a parallel interface of independently controlled serial links.

PCI-X

An enhanced PCI bus technology that is backward compatible with existing PCI cards but with increased bandwidth. PCI-X runs at 100 MHz or 133 MHz.

Plug and Play (PnP)

A set of specifications with firmware and operating-system capability that automatically configures a computer system and its peripheral devices. PnP also specifies a set of interfaces for device drivers.

RCIM

Real-Time Clock and Interrupt Module. A multifunction PCI card designed by Concurrent for fully deterministic event synchronization in multiple CPU applications. The RCIM includes a synchronized clock, multiple programmable real-time clocks, and multiple input and output external interrupt lines. Interrupts can be shared (distributed) across interconnected systems using an RCIM chain.

slot priority

The hardware priority used in the system for all maskable hardware interrupts belonging only to PCI cards installed in the system.

SNMP

Simple Network Management Protocol – a standard interface that allows a network manager to remotely monitor and manage workstations.

SVGA

Super video graphics array. VGA and SVGA are video standards for video adapters with greater resolution and color display capabilities than previous standards.

system priority

The hardware priority used in the system for all maskable and non-maskable hardware interrupts.

USB

Universal Serial Bus – a USB connector provides a single connection point for multiple USB-compliant devices, such as mice and keyboards. USB devices can be connected and disconnected while the system is running.

vector

An integer from 0-255 that is used to refer to an interrupt service routine, or branch instruction to an interrupt service routine. The vector is stored in an IOAPIC register.

VGA

Video graphics array. VGA and SVGA are video standards for video adapters with greater resolution and color display capabilities than previous standards.

Index