December 1996

Compaq Computer Corporation

Where Do I Plug the Cable? Solving the Logical-Physical Slot Numbering Problem

CONTENTS

ABSTRACT

As the number of identical PCI devices performing unique functions in the same server increases, it becomes increasingly difficult to identify the physical location of a specific PCI device. This paper will explain the need for communicating to the System Administrator the unique physical location of each PCI device, specifically the chassis and slot numbers. New PCI-to-PCI bridge registers designed to help solve this problem and defined in the upcoming revision to the *PCI-to-PCI Bridge Architecture Specification* are described. The algorithm for a Compaq proposed function call is also presented. The function uses the new bridge registers to convert from logical bus and device number to physical chassis and slot number.

Please direct comments regarding this communication via internal bmail to this address:
Tech Com@HW Tech@Sys Hou.
Compaq field personnel may also send comments via the internet to this address:
tech_com@bangate.compaq.com

COMPAQ

NOTICE

The information in this publication is subject to change without notice.

COMPAQ COMPUTER CORPORATION SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL. THIS INFORMATION IS PROVIDED "AS IS." COMPAQ MAKES NO REPRESENTATION OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, OR STATUTORY, AND EXPRESSLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSE, GOOD TITLE AND NONINFRINGEMENT.

This publication does not constitute an endorsement of the product or products that were tested. The configuration or configurations tested or described may or may not be the only available solution. This test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state or local requirements. Compaq does not warrant products other than its own strictly as stated in Compaq product warranties.

Compaq, Contura, Deskpro, Fastart, Compaq Insight Manager, LTE, PageMarq, Systempro, Systempro/LT, ProLiant, TwinTray, ROMPaq, LicensePaq, QVision, SLT, ProLinea, SmartStart, NetFlex, DirectPlus, QuickFind, RemotePaq, BackPaq, TechPaq, SpeedPaq, QuickBack, PaqFax, Presario, SilentCool, CompaqCare (design), Aero, SmartStation, MiniStation, and PaqRap, registered United States Patent and Trademark Office.

Netelligent, Smart Uplink, Extended Repeater Architecture, Scalable Clock Architecture, Armada, Cruiser, Concerto, QuickChoice, ProSignia, Systempro/XL, Net1, LTE Elite, Vocalyst, PageMate, SoftPaq, FirstPaq, SolutionPaq, EasyPoint, EZ Help, MaxLight, MultiLock, QuickBlank, QuickLock, UltraView, Innovate logo, Wonder Tools logo in black/white and color, and Compaq PC Card Solution logo are trademarks and/or service marks of Compaq Computer Corporation.

Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

©1996 Compaq Computer Corporation. All rights reserved. Printed in the U.S.A.

Microsoft, Windows, Windows NT, Windows NT Advanced Server, SQL Server for Windows NT are trademarks and/or registered trademarks of Microsoft Corporation.

Where Do I Plug the Cable?
Solving the Logical-Physical Slot Numbering Problem
First Edition (December 1996)
209A/1296

INTRODUCTION

As the number of identical PCI devices performing unique functions in a server grows, identifying the physical location of a specific PCI device becomes increasingly difficult. Compaq has taken the lead in solving this technology problem. A Compaq proposal for a general solution has been under review in the PCI community since mid-1995, and Compaq has been working with the PCI SIG to provide support for an industry-wide solution in PCI-to-PCI bridges.

This brief explains the requirement for slot and chassis number identification in servers and PCI expansion systems. It discusses the new PCI-to-PCI bridge registers defined in the upcoming revision to the PCI-to-PCI Bridge Architecture Specification and the algorithm for the function call Compaq has proposed.

THE NEED FOR SLOT NUMBERS

The PCI standard has been able to deliver on the "plug and play" promise by requiring that any compliant device be able to accept any valid resource configuration at power-up. PCI BIOS assigns resources at power-up, automatically allocating system resources without conflict. Unlike previous standards, however, the PCI standard does not include the concept of a "slot," that is, a description of the physical location of a device within the system. This has been caused, primarily by the desktop heritage of PCI Technology development.

Desktop Computers

In a standard desktop computer, there are usually few PCI expansion slots and rarely more than one instance of any device type. This makes it easy to identify the physical location (the slot) of any particular device. The typical desktop includes slots for a graphic controller, a network controller, and mass storage controller. Since each device has only one connector and the shapes and sizes of connectors differ, attaching the cables to the appropriate device is straightforward (Figure 1). In desktops with PCI slots the configuration process executes each time the machine powers on, so resource conflicts do not occur, even after a controller has been exchanged or a new controller has been added.

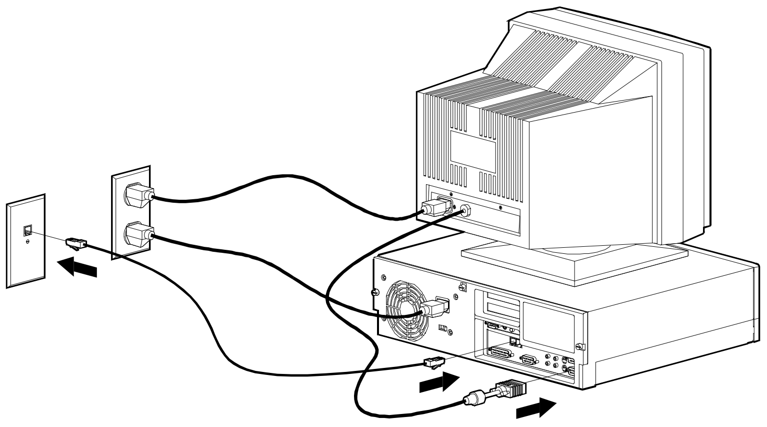


Figure 1. Desktop computer applications typically have few external connections and no duplicate connectors, so connecting cables is straightforward. This one has only a video monitor and LAN.

Servers

Identifying a particular physical device becomes confusing with PCI-based network servers. A PCI-based server typically contains a large number of PCI expansion slots. In mid-1996 servers averaged six to eight slots. These expansion slots are likely to be filled with multiple, sometimes identical controller-types that provide support for network segments and multiple disk channels. Servers containing four or more disk controllers or five network controllers are not uncommon, especially in large database configurations. While one disk controller may connect to a number of SCSI disks, another may control multiple tape drives. One network controller may connect to hundreds of systems in an office building, while another network controller may handle a connection to the Internet (Figure 2).

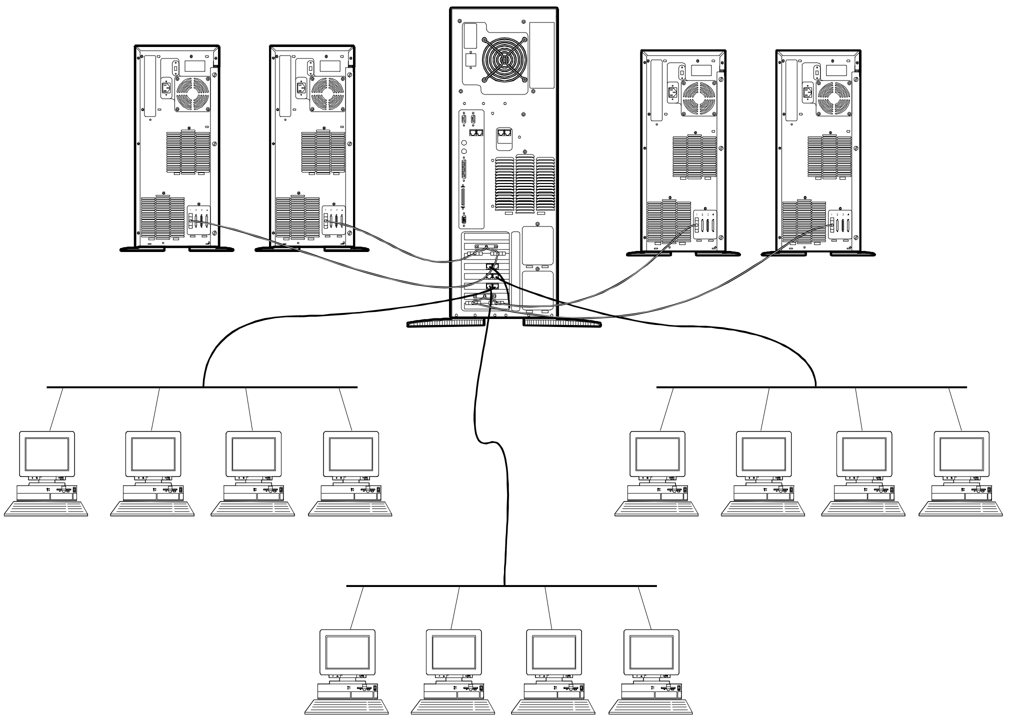


Figure 2. A typical server (center) may have identical electrical connections to multiple storage subsystems (across the top), and multiple identical electrical connections to different LAN segments (bottom).

Through PCI and other system functions, the user no longer needs to allocate interrupts or memory address ranges manually. However, there are still situations in which a user must identify a particular controller both logically and physically. For example:

1. When plugging in an external cable, the user must identify the correct connector.
2. When configuring items such as the operating system, device drivers, and protocol stacks, the software will require a way to identify the device. For example, when configuring network controllers, the user must typically specify a controller (identified by slot number), and then assign a network address and the protocols to use with that controller.
3. When a controller of any type fails in a system, software such as diagnostic tools must have a way to communicate which controller has failed so that the user can replace it.

Consider the following example: A PCI server with two identical Ethernet controllers has one controller cabled to a small number of workstations. The other controller is cabled to an Ethernet backbone that runs throughout the company. To configure all the software running on the server properly, the two network controllers must be assigned TCP/IP addresses. The user must match a software configuration parameter (the IP address) to a piece of hardware (one of the two controllers). Without a unique identifier such as a slot number, the user has no constant identifier that is guaranteed to remain the same no matter how the rest of the system is reconfigured.

The system software can, of course, uniquely identify each controller *logically* by a PCI bus number and device number. The problem is presenting this information to the user, so that the user can *physically* locate the controller.

Why Can't the User Use Bus and Device Number?

Although the PCI bus number and device number do uniquely identify each controller, these identifiers fall short of the user's needs in two areas. First, the slot number is a familiar paradigm for users. Users already understand the concept of "slot number." Instructing a user to install a controller "at bus 0, device 4" would require a shift in the user's thought process. The slot number provides an intuitive method for the user to physically identify a controller.

But more importantly, using PCI bus numbers and device numbers as an identification method is deficient for another reason: PCI bus numbers do not necessarily remain constant. When multiple host-to-PCI bridges or PCI-to-PCI bridges are embedded in the system, there are multiple buses to be numbered. The numbers assigned to the buses can change when the system is reconfigured. Because bus numbers are assigned during the boot process, just like other system resources, there is no guarantee that they will remain constant across boot cycles. Thus, if the user configures software to use a controller found on "bus number 2, device number 7," and later adds another controller that happens to have its own PCI bus embedded, then any bus number beyond bus 0 will potentially be reassigned. The reassignments are based on the location of the controller and on the order in which the system's BIOS finds and configures PCI devices during the boot process. If bus number 2 is reassigned to bus number 3, then the user's software configuration will be incorrect, as will any slot markings or configuration notes he may have made to help locate the device. A physical identifier, such as a slot number, remains constant across boot cycles and therefore provides a better solution to the problem.

SLOT NUMBERS IN THE IRQ ROUTING TABLE

These types of challenges brought about the creation of the IRQ Routing Table call in the PCI BIOS, which was added to the *PCI BIOS Specification*, Revision 2.1. Using the bus number and device number, software can perform a table lookup to retrieve information about how each device in the main chassis is wired. One of the fields defined in the IRQ Routing Table is the device slot number. The device slot number is defined in the IRQ Routing Table, as shown in Figure 3. The software uses the information in the IRQ Routing Table to translate the physical slot number into PCI bus number and device number for devices in the main chassis.

Offset	Size	Field Description	Value
0	byte	PCI Bus Number	00h
1	byte	PCI Device Number (in upper 5 bits)	58h
2	byte	Link value for INTA#	3
3	word	IRQ bit-map for INTA#	0FFFFh
5	byte	Link value for INTB#	4
6	word	IRQ bit-map for INTB#	0FFFFh
8	byte	Link value for INTC#	3
9	word	IRQ bit-map for INTC#	0FFFFh
11	byte	Link value for INTD#	4
12	word	IRQ bit-map for INTD#	0FFFFh
14	byte	**Physical Slot Number**	**5**
15	byte	Reserved	0
16	byte	PCI Bus Number	00h
17	byte	PCI Device Number (in upper 5 bits)	70h
18	byte	Link value for INTA#	5
19	word	IRQ bit-map for INTA#	0FFFFh
21	byte	Link value for INTB#	6
22	word	IRQ bit-map for INTB#	0FFFFh
24	byte	Link value for INTC#	5
25	word	IRQ bit-map for INTC#	0FFFFh
27	byte	Link value for INTD#	6
28	word	IRQ bit-map for INTD#	0FFFFh
30	byte	**Physical Slot Number**	**6**
31	byte	Reserved	0
32..xx		*Additional PCI Device Entries*	

Figure 3. An excerpt from a typical IRQ Routing Table defining how PCI interrupts are connected for devices in the main chassis. This table can be used to translate from PCI bus and device number to slot number for devices in the main chassis.

PCI EXPANSION SYSTEMS

A PCI expansion system can be described as PCI expansion slots housed in an external cabinet and connected to a server through one or more PCI-to-PCI bridges, as shown in Figure 4. Expansion systems are a recent addition to the PCI product landscape, because they are only useful in server environments in which large numbers of I/O controllers (disk controllers, network controllers and communications controllers) are used. A network file server may require these expansion cabinets when all PCI slots in the server are already in use.

Expansion cabinets complicate the problem of physically locating a device. Not only does the user need to locate a connector in a specific slot, he must also search multiple external cabinets for the controller. Furthermore, slots in expansion cabinets cannot be included in the IRQ Routing Table because the BIOS has no way of determining what expansion system might be installed by the user.

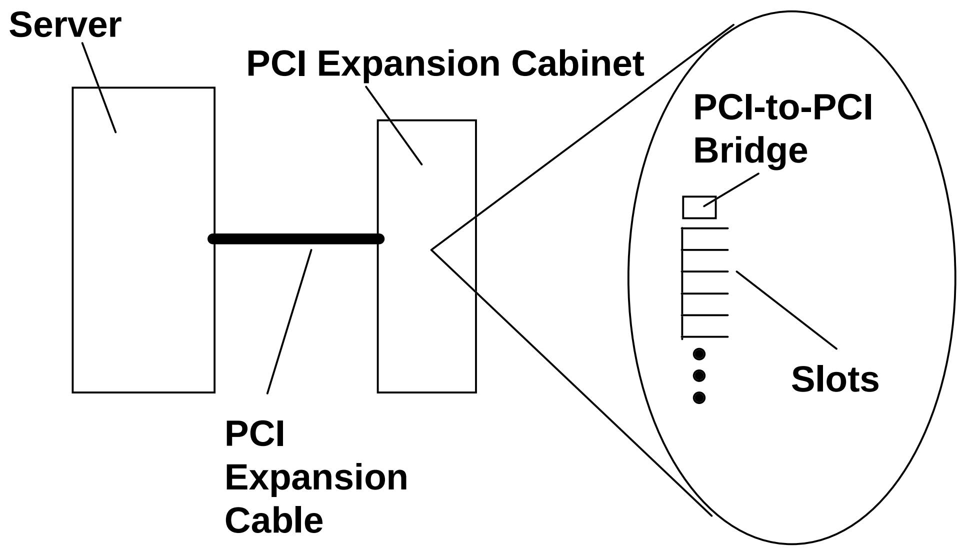


Figure 4. In a PCI expansion system , additional PCI slots are provided in a separate cabinet, further complicating the problem of unique physical identification of a device. The slot numbering proposal assigns a unique "chassis number" to each cabinet.

A COMPREHENSIVE SLOT NUMBERING PROPOSAL

Since the IRQ Routing Table solves the slot numbering problem in the main chassis, what is required is a standard method for determining slot number in a PCI expansion system. In mid-1995, Compaq Computer Corporation began circulating for review within the PCI community a proposal for a general solution to this problem. The hardware required to support this proposal is being included in Revision 1.1 of the *PCI-to-PCI Bridge Architecture Specification*. The following discussion presents the hardware aspects of the proposal and then the software aspects.

The New Registers

If we assume that the gateway to an expansion system is always a PCI-to-PCI bridge, then the logical place to define a standard solution to the slot numbering problem for expansion systems is the bridge. The two new registers shown in Figure 5, the Chassis Number register and the Expansion Slot register, provide the necessary information to make the conversion from device number to slot number. The PCI-to-PCI Bridge Architecture Specification will document these two new registers as optionally implemented (required only for bridges intended for use with expansion chassis), and determine a standard location for them, possibly using the same "Extended Capabilities" mechanism currently being considered by the PCI SIG for Type 0 configuration space.

15 8	7 0
Chassis Number	Expansion Slot

Figure 5. The two newly defined registers for slot numbering are located in a newly defined extension to the standard Configuration Space Header for PCI-to-PCI bridges.

The host system is assigned chassis number 0. Each external cabinet that contains PCI slots is assigned another unique chassis number. The new Chassis Number register in the PCI-to-PCI bridge contains a single 8-bit number that designates the chassis in which the slots on the bridge's secondary bus reside. Multiple PCI buses contained in the same chassis should be assigned the same chassis number.

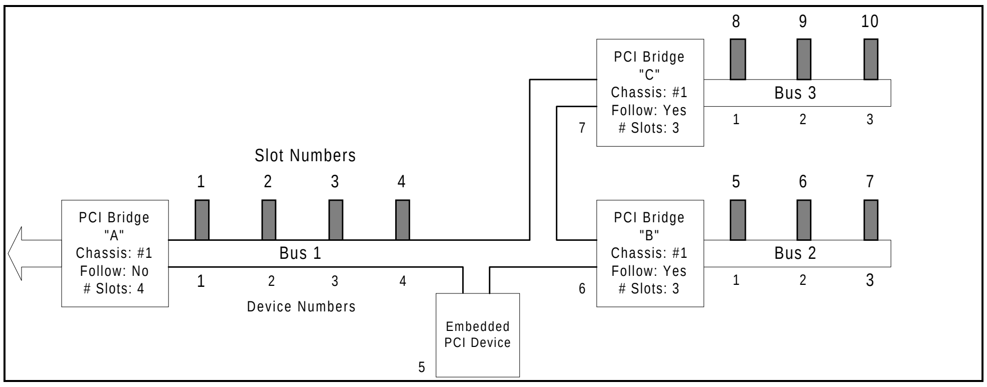
The Chassis Number register can be initialized either by the power-up system configuration software or by hardware. If the register is to be initialized by software, then the register will be read-write. It can be non-volatile or it can be initialized to 0 at power-up. If software determines that the register is read-write and that the value is 0 or equals another chassis' number, then software will assign a new chassis number. If the register is initialized by hardware, then the register will be read-only, and the system designer must provide a means for the user to change the chassis number if there is a conflict.

The details of the Expansion Slot register are shown in Figure 6. Bits 4 through 0 of the Expansion Slot Provided field contains the binary encoded value of the number of expansion slots that are provided directly on the secondary bus of this bridge. If no expansion slots are implemented behind a particular bridge, then this register should be initialized to 0.

7 6	5	4 3 2 1 0	
Reserved	Slots Follow Parent	Expansion Slots Provided	F0h

Figure 6. Expansion Slot register. The information encoded in this register includes the number of expansion slots provided directly behind this bridge and the Slots Follow Parent bit that indicates whether multiple bridges with expansion slots are cascaded within one chassis.

To understand how the Slots Follow Parent bit is used, it is first necessary to consider how PCI expansion systems might be configured. Figure 7 illustrates one such system. The bridge that controls the first slot in the expansion chassis (Bridge A) is referred to as the "parent" bridge. Its Slots Follow Parent bit is set to 0 (no) to indicate that it is the parent. Additional bridges whose slot numbers follow the parent slots (Bridges B and C) are referred to as "child" bridges, and their Slots Follow Parent bits will be set to 1 (yes).



8
9
10
PCI Bridge "C" Chassis: #1 Follow: Yes # Slots: 3
Bus 3
1
2
3
7
Slot Numbers
1
2
3
4
5
6
7
PCI Bridge "A" Chassis: #1 Follow: No # Slots: 4
Bus 1
1
2
3
4
Device Numbers
PCI Bridge "B" Chassis: #1 Follow: Yes # Slots: 3
Bus 2
1
2
3
6
Embedded PCI Device
5

Figure 7. PCI expansion chassis containing a hierarchy of bridges and devices. Bridge A is the "parent" since its slots come first and its Slots Follow Parent bit is reset. Bridges B and C are "children" since their slots number sequentially after Bridge A, and their Slots Follow Parent bit is set. Bridge B's device number must come before Bridge C's since Bridge B's slots number first.

Because the Expansion Slot Register provides the power-up system configuration software with vital information about the physical arrangement of the system, this register must be initialized before the power-up system configuration software runs. This generally implies that the Expansion Slot register must be initialized by hardware. The means by which the system designer programs this information into the hardware is not specified, and is, therefore, left to the creativity of the bridge designer. The simplest approach would be to initialize the register contents with the state of certain device package pins at RST# time. However, more elaborate schemes involving shift registers or even serial EEPROMs could reduce pin count or provide more flexibility and convenience to the user at the cost of increased hardware complexity.

Finding Chassis and Slot Number

Chassis numbers are established by the system configuration software each time the system is reconfigured. The host system chassis is always chassis 0, and expansion chassis numbers are stored in the Chassis Number registers in the appropriate bridges. After the system has been initialized, any software needing the chassis number for a device can first check the IRQ Routing Table to determine whether the device is in chassis 0. If not, the software must then find the bridge whose secondary bus number matches the bus number of the device in question. If this bridge supports expansion slots, then the chassis number can be read directly from the Chassis Number register. If this bridge does not support expansion slots, (an embedded bridge), then the chassis number is read from the bridge which supports the slot in which the embedded bridge is installed.

The slot number of a device in the main chassis can be found just as simply as the chassis number by looking in the IRQ Routing Table. However, in an expansion chassis the slot number of a device must be calculated from the device number and Slots Provided register. The following assumptions are made to calculate the slot number for a device in a PCI expansion system:

1. Slot numbers within each expansion chassis start at 1 and increment sequentially.
2. The PCI device number for each expansion slot starts at 1 and increments sequentially.
3. If an expansion system has multiple child bridges with the same parent bridge, then the child bridge with the lower slot numbers must also have the lower device number on the parent bus.

To calculate the slot number for a device in an expansion chassis, the software must first find the bridge whose secondary bus number matches the device's bus number. If the Slots Follow Parent bit is *not* set in this bridge (a parent bridge), then the slot number is equal to the PCI device number. If the Slots Follow Parent bit *is* set in this bridge (a child bridge), the software calculates the slot number by adding the following three numbers:

1. Device number for this device.
2. Value from the Expansion Slots Provided field from the *parent* bridge.
3. Value from the Expansion Slots Provided field from all *other child* bridges of this parent, whose device numbers are less than the device number of *this* child bridge. (See Figure 7.)

If the slot numbering algorithm encounters a bridge that does not support the Chassis Number and Expansion Slot registers, then it assumes that there are no expansion slots behind that bridge. All devices behind that bridge will inherit the same slot number as the bridge itself. In this way a card such as a multi-headed NIC or SCSI controller will report the same slot number for all devices on that card.

Figures 8 and 9 illustrate an algorithm of finding chassis and slot numbers for devices in an expansion chassis. The algorithm starts at the top of the configuration hierarchy and scans every device, accumulating chassis and slot information until the designated device is encountered.

A Slot Numbering Example

The diagram shown in Figure 7 contains all the elements that can affect the numbering of PCI expansion slots. It represents a single, external expansion chassis, which would be connected to the system via the PCI-to-PCI bridge on the left side (the arrow indicates the connection to the system). Above each expansion slot is the Slot Number that would be physically labeled on the slot. The other numbers shown are the PCI Device Numbers that would be assigned to each (potential) device in the chassis.

The first PCI-to-PCI bridge (left side of the diagram) has four PCI expansion slots on its secondary interface (Bus 1). Since the Slots Follow Parent field (labeled "Follow" in the diagram) is not set, these slots must be the first slots within the chassis. They are therefore numbered 1 through 4. Also on Bus 1 is an embedded PCI device located at Device Number 5.

At Device Number 6 is a PCI-to-PCI bridge, which reports three expansion slots on its secondary interface. This child bridge reports the same Chassis Number as the parent bridge, and its slots should follow those of the parent bridge. Since this bridge is the lowest numbered bridge device on Bus 1, its slots follow the parent bridge before higher-device numbered bridges. Therefore, its slots are numbered 5, 6, and 7.

The final device on Bus 1, Device Number 7, holds another PCI-to-PCI bridge, that also reports three expansion slots. Because its slots follow the parent bridge, the slots are numbered 8, 9, and 10.

The "Find PCI Slot Number" Function Call

The slot numbering proposal includes the addition of a "Find PCI Slot Number" function call to simplify the conversion between bus-device numbers and chassis-slot numbers. This function call would be implemented by the Operating System, making it available to any device driver or other program that requires it.

"Find PCI Slot Number" uses the IRQ Routing Table to find slot numbers in the main chassis (chassis 0) and uses the algorithm in Figures 8 and 9 to find numbers in the expansion chassis.

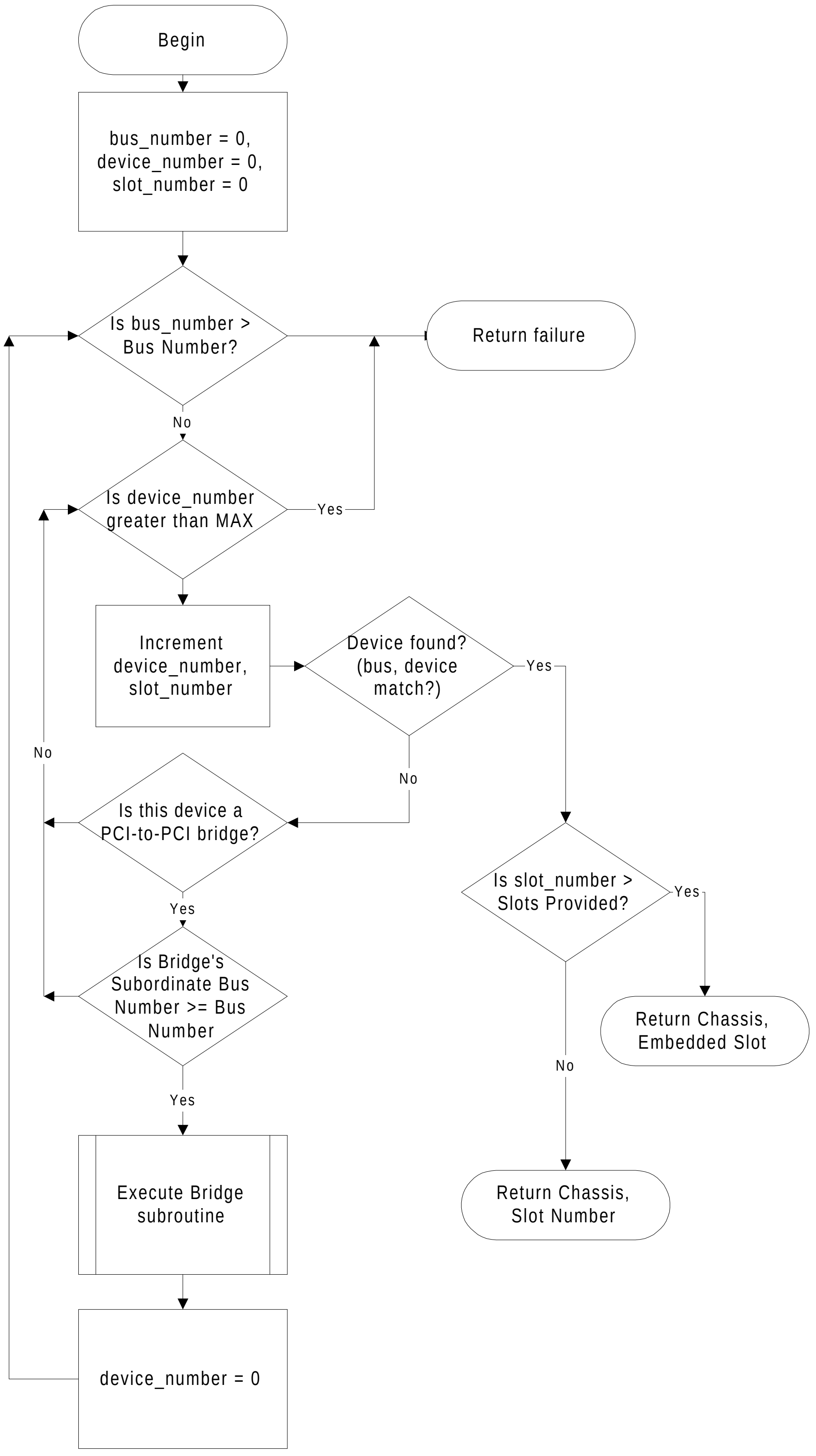

Begin
bus_number = 0, device_number = 0, slot_number = 0
Is bus_number > Bus Number?
Return failure
No
Is device_number greater than MAX
Yes
Increment device_number, slot_number
Device found? (bus, device match?)
Yes
No
No
Is this device a PCI-to-PCI bridge?
Is slot_number > Slots Provided?
Yes
Yes
Is Bridge's Subordinate Bus Number >= Bus Number
Return Chassis, Embedded Slot
No
Yes
Execute Bridge subroutine
Return Chassis, Slot Number
device_number = 0

Figure 8. Flowchart for "Find PCI Slot Number" Function Call.

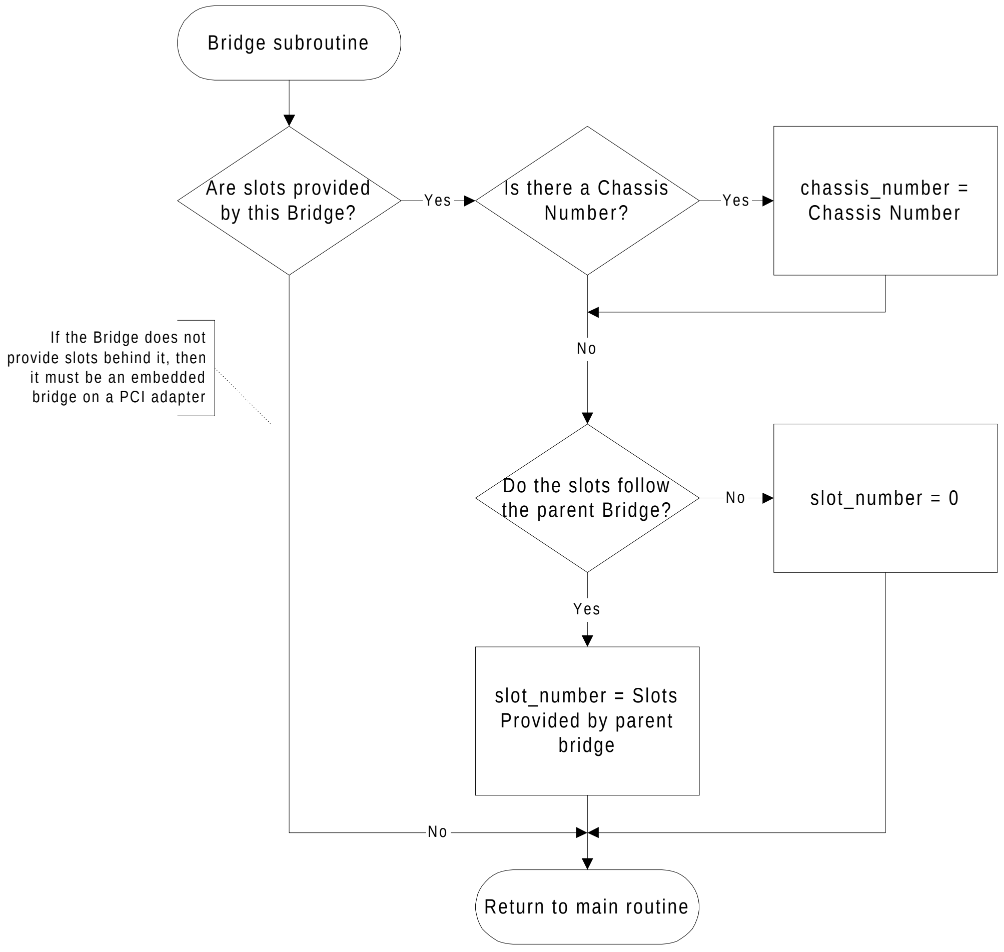

Bridge subroutine
Are slots provided by this Bridge?
Yes
Is there a Chassis Number?
Yes
chassis_number = Chassis Number
No
If the Bridge does not provide slots behind it, then it must be an embedded bridge on a PCI adapter
Do the slots follow the parent Bridge?
No
slot_number = 0
Yes
slot_number = Slots Provided by parent bridge
No
Return to main routine

Figure 9. Flowchart for "Bridge Found" subroutine for the "Find PCI Slot Number" Function Call.

THE FUTURE OF PCI SLOT NUMBERING

As mentioned previously, the hardware necessary to support this proposal for the PCI expansion chassis is being added to version 1.1 of the *PCI-to-PCI Bridge Architecture Specification*. This same revision also specifies how Delayed Transactions work with PCI bridges. Compaq is encouraging multiple PCI-to-PCI bridge vendors to include the slot numbering register in their next generation bridge designs, even before the new revision of the bridge specification is released. Compaq strongly encourages designers of PCI expansion chassis to select a bridge that includes the hardware necessary to support PCI slot numbering.

Now that the hardware support is being implemented, support for a new function call needs to be added. Compaq has already begun circulating a proposal within the user communities for review and comment. The proposal would add a "Find PCI Slot Number" function call to the next revision of an Operating System.. Although the new hardware for slot numbering can be used without the new function call, the inclusion of the new call will simplify the delivery of an accurate implementation of the algorithm, especially in areas such as device drivers, diagnostics and system management utilities. Compaq expects that advanced operating systems will not wait for the introduction of the hardware registers for the expansion chassis. These OS vendors will begin immediately to implement the algorithm described in this brief. Since slot numbers were added to the IRQ routing table in the August 1994 release of version 2.1 of the PCI BIOS Specification, solutions for the main chassis will already work. When expansion chassis with the new bridge registers become available, numbering slots in the expansion chassis will work, too. Device driver and application writers should watch closely for developments from their OS vendor.

SUMMARY

As the number of PCI slots grows to accommodate multiple identical controllers performing unique functions, it has become difficult to identify the physical location of a particular controller. The inclusion of the slot number in the PCI BIOS IRQ Routing Table and of the Chassis Number and Expansion Slot registers in new PCI-to-PCI bridge implementations will enable the translation from the logical bus and device number to the physical chassis and slot number for all controllers in the system.