Introduction 1

This architecture specification provides a comprehensive computer system hardware-to-software interface definition, combined with minimum system requirements, that enables the development of and software porting to a range of compatible industry-standard computer systems from portables through servers. These systems are based on the PowerPC microprocessor, as defined in *The PowerPC Architecture* [1]. The definition supports the development of both uniprocessor and multiprocessor system implementations.

A key attribute and benefit of the architecture is the ability of hardware platform developers to have degrees of freedom of implementation below the level of architected interfaces and therefore have the opportunity for adding unique value. This flexibility is achieved through architecture facilities including: (1) device drivers; (2) Open Firmware (OF); (3) Run-Time Abstraction Services (RTAS); and (4) hardware abstraction layers. The role of items 1 and 4 are described in separate operating system documentation. The role that items 2 and 3 play in the architecture will be described in subsequent paragraphs and chapters.

Though the PowerPC microprocessor is the most widely used RISC processor, substantial legacy software exists and a mechanism for running the bulk of this legacy software is a requirement. The system address map has been defined with a specific objective of assisting efficient x86 emulation. Additionally, the PowerPC microprocessors support Bi-Endian (see Section 2.3, “Bi-Endian Support,” on page 14 and Appendix C, “Bi-Endian Designs,” on page 265) operation which is a key attribute important to running the supported operating systems and applications. Bi-Endian capability is not available in the current IBM PC compatible x86-based system architecture.

The architecture combines leading-edge IBM PC and Apple Macintosh technologies to create a superior personal computing platform. By design, it

supports a wide range of computing needs including personal productivity, engineering design, data management, information analysis, education, desktop publishing, multimedia, entertainment, and database, file, and application servers. The architecture effectively leverages industry-standard I/O through the PCI bus while accommodating legacy I/O from both the IBM PC compatible and the Apple Macintosh domains. This approach provides several key benefits for system manufacturers and end customers: (1) systems can be designed and manufactured to enable the customer a choice of operating system support which could include AIX, Mac OS, NetWare, OS/2, Solaris, or Windows NT and (2) smooth application, operating system, and customer system transitions are enabled by accommodation for legacy software, I/O devices, and peripherals. This architecture helps protect the customer's investment while moving to more advanced portable, desktop, and server computing platforms. Systems based on this architecture are expected to offer price/performance advantages and to address the expected growth in computing performance and functionality.

CHRP systems must be able to meet certain minimum requirements to be considered compliant with the architecture specified in this document. Apple, IBM, and Motorola are reviewing these requirements and intend to make them available at a later date.

1.1 Platform Topology

To the experienced computer designer and system manufacturer, much of the content of the architecture will be familiar. A typical desktop topology is shown in Figure 1 on page 4. This topology consists of a single PowerPC microprocessor, volatile System Memory, and a single Host Bridge providing a PCI Bus. The PCI Bus and ISA Bridge provide for connection of I/O devices.

A more complex general topology is shown in Figure 2 on page 5. All platforms consist of one or more PowerPC microprocessors, a volatile System Memory separate from other subsystems, and a number of I/O devices, which may initiate transactions to System Memory. The processors are linked over the primary processor bus/switch to each other, to the System Memory, and to one or more Host Bridges (Host Bridge 0 must be a PCI Host Bridge). In general, I/O devices do not connect to the primary processor bus/switch. The Host Bridges connect to secondary buses which have I/O devices connected to them. In turn, one or more bus bridges may be employed to tertiary buses (for instance ISA or PCI) with additional I/O devices connected to them. Typically, the bus speeds and throughput decrease and the number of supportable loads increases as one progresses from the primary processor bus to more remote buses. Multiprocessor platforms have a symmetric (at least from the software

Personal Use Copy - Not for Reproduction

point of view) shared memory model; refer to Section 12.1, “SMP System Organization,” on page 216.

There are variations on these topologies, which are likely to occur and are therefore worth describing below. The architecture describes interfaces, not implementation. The logical software model must remain the same, even if the physical topology is different.

- In a smaller platform, the Host Bridge and/or the memory and/or an I/O device may be integrated into a single chip. In this case, the topology would not look like Figure 1 from a chip point of view, but instead would be integrated onto the single chip.
- In a larger platform, secondary buses may be implemented, with two or more Host Bridges, as two or more parallel expansion buses for performance reasons. Similarly, tertiary buses may be two or more parallel expansion buses off each secondary bus. This is indicated by the ellipses near the Host Bridge and the Bus Bridge.
- In a high performance platform, with multiple processors and multiple memories, a switch may be employed to allow multiple parallel accesses by the processors to memory. The path through the switches would be decided by the addressing of the memory.

Each of the following chapters provides information necessary to successfully implement compliant systems. It is recommended that the reader become thoroughly familiar with the contents of these chapters and their references prior to beginning system or software development. It is anticipated that standard chip sets will simplify the development of compliant implementations consistent with the topologies shown below and will be available from third-party industry sources.

Personal Use Copy - Not for Reproduction

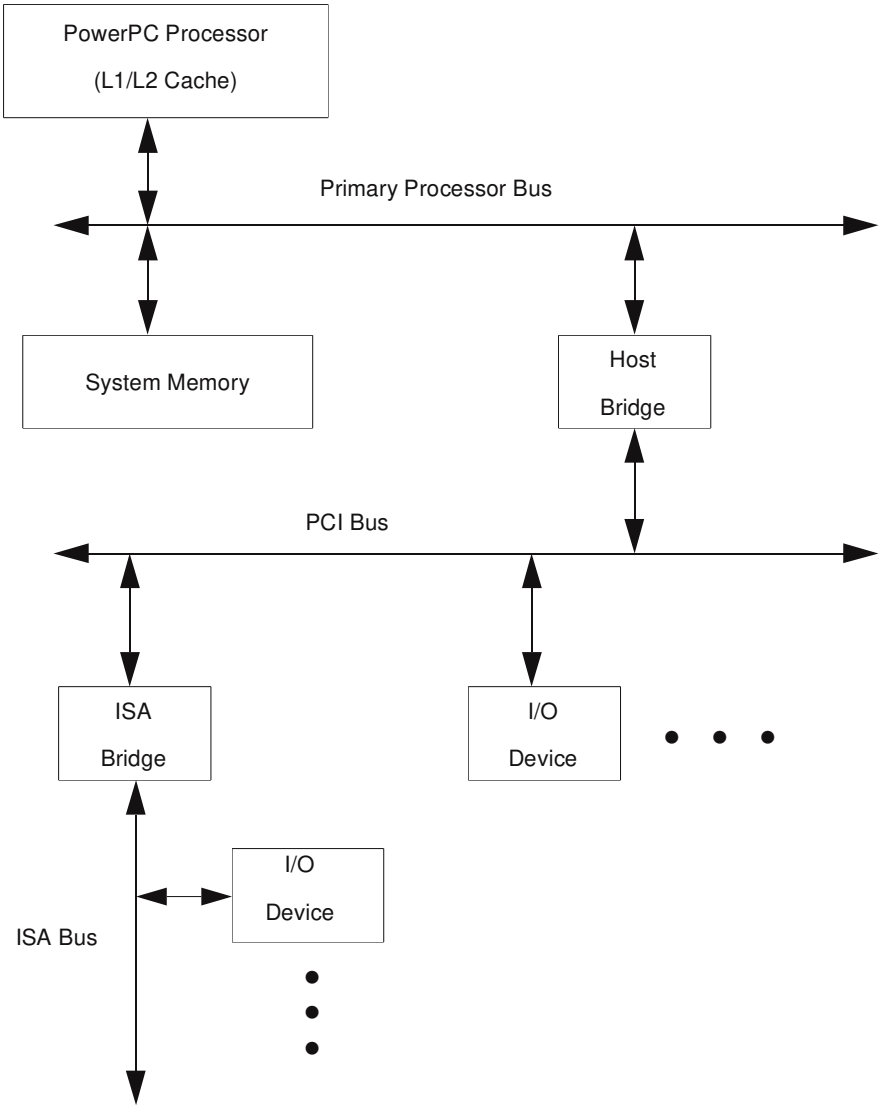


Figure 1. Typical Desktop Topology

Personal Use Copy - Not for Reproduction

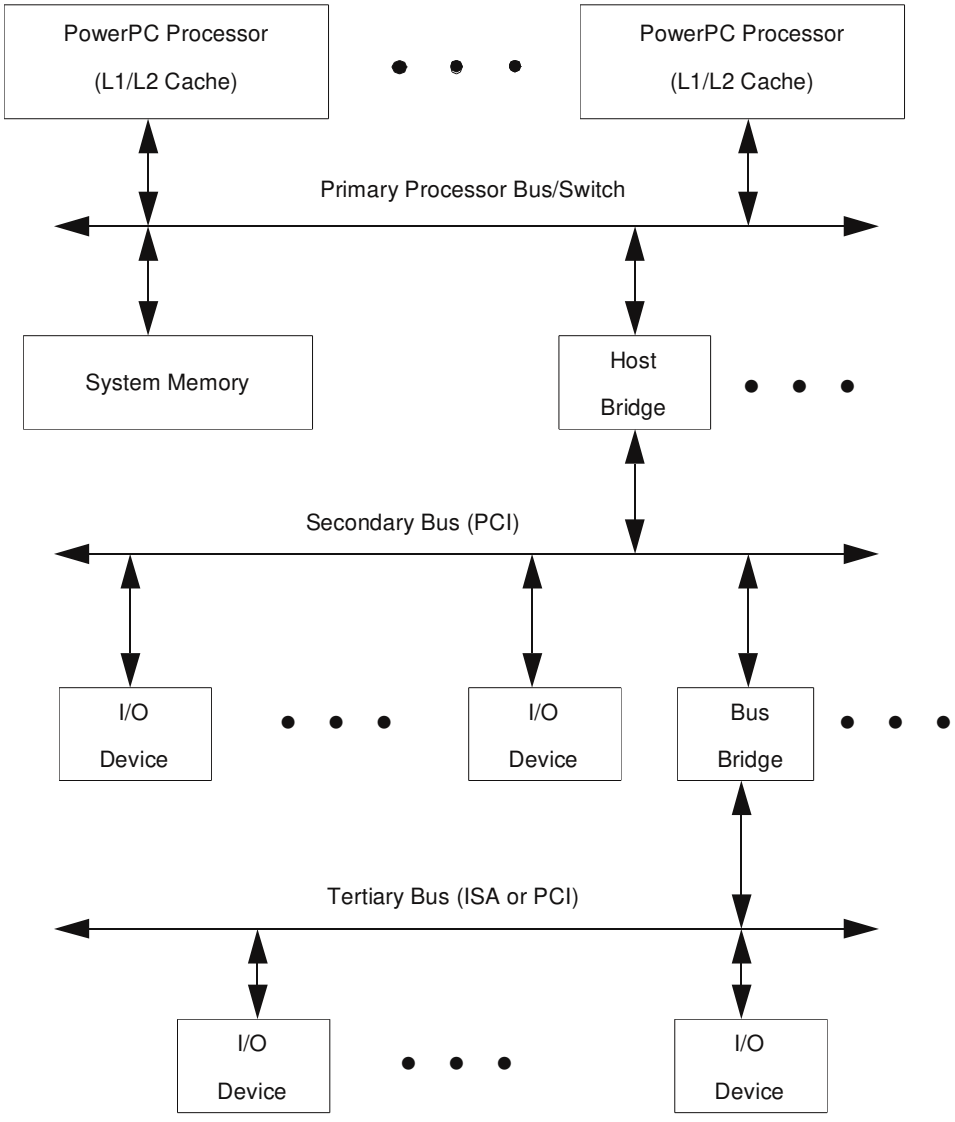


Figure 2. General Platform Topology

Personal Use Copy - Not for Reproduction

Blank page when cut - 6

System Requirements

2

This chapter gives an operational overview of Common Hardware Reference Platform systems and introduces platform specific software components that are required for operating system support. The chapter also addresses some system level requirements that are broad in nature and are fundamental to the architecture described in later chapters. Lastly, a table of requirements is presented as a guide for platform providers.

2.1 System Operation

2.1.1 Control Flow

Figure 3 on page 8 is an example of typical phases of operation from power-on to full system operation to termination. This section gives an overview of the processes involved in moving through these phases of operation. This section will introduce concepts and terms that will be explained in more detail in the following chapters. Most requirements relating to these processes will also appear in later chapters.

The discussion in this chapter will be restricted to systems with a single processor. Refer to Chapter 12, “The Symmetric Multiprocessor Option,” on page 215 for the unique requirements relating to multiprocessor systems.

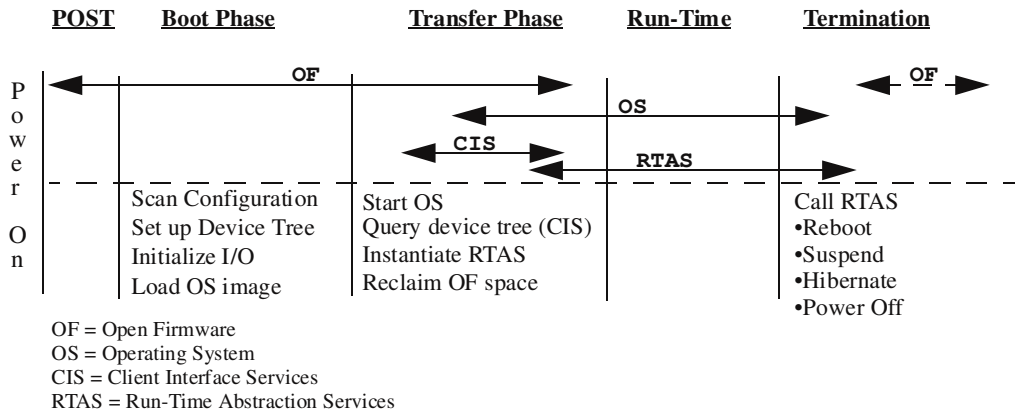


Figure 3. Phases of Operation (example)

2.1.2 POST

Power On Self Test (POST) is the process by which the firmware tests those areas of the hardware that are critical to its ability to carry out the boot process. It is not intended to be all-inclusive or to be sophisticated in how it relates to the user. Diagnostics with these characteristics will generally be provided as a service aid.

2.1.3 Boot Phase

The following sections describe the boot phase of operation. The fundamental requirements of the boot phase are:

1. Identify and configure system components.
2. Generate a Device Tree.
3. Initialize/reset system components.
4. Locate an operating system boot image.
5. Load the boot image into memory.

Personal Use Copy - Not for Reproduction

2.1.3.1 Identify and Configure System Components

The firmware must, by various means, become aware of every component in the system and configure or reset those components into a known state. Components include all bridges and device controllers, but may exclude devices that are not involved in the boot process.

Firmware is generally written with a hardware platform in mind, so that some components and their configuration data can be hardcoded. Examples of these components are: number and type of processors, cache characteristics, and the use of components on the planar. This hardcoding is not a requirement, only a practical approach to a part of this task.

Certain system information must come from "walking" the I/O buses. This is a technique that will yield identification of I/O device controllers and bridges that reside on modern, well-behaved buses such as PCI. In general, it is not possible to walk the ISA bus.

ISA configuration requires either Plug-n-Play (PnP) protocols and adapters or special handling by the system firmware or software. This architecture requires a portion of non-volatile memory (name = *isa-config*) for storage of ISA configuration data which allows firmware and operating systems to record and retrieve this data as required (see Section 8.4.4, "Configuration (0x71)," on page 145).

2.1.3.2 Generate a Device Tree

The firmware will build a device tree. The operating system will gain access to the device tree through Client Interface Services (CIS).

Certain configuration information (configuration variables) may be stored in non-volatile memory. They will be stored under the partition names *of-config* or *common,* depending on the nature of the information (see Chapter 8, "Non-Volatile Memory," on page 141).

2.1.3.3 Initialize/Reset System Components

Operating Systems require devices to be in a known state at the time control is transferred from the firmware. Firmware may gain control with the hardware in various states depending on what has initiated the boot process.

- Normal boot: Initiated by a power-on sequence; all devices and registers begin in a hardware reset state.
- Wakeup from Hibernation: Should be indistinguishable from a normal boot.

Personal Use Copy - Not for Reproduction

- Resume from Suspend: Devices that were left powered on during suspend may be in the state left by software or in a hardware reset state. Firmware must be able to deal with this and alter the state as necessary to conform to the conditions below.
- Reboot: Device state is unpredictable. Firmware must be able to deal with this and alter the state as necessary to conform to the conditions below.

The hardware reset state for a device is an *inactive* state. An *inactive* state is defined as a state that allows no system level activity; there can be no bus activity, interrupt requests, or DMA requests possible from a device that is in a reset state. Since operating systems may configure devices in a manner that requires very specific control over these functions to avoid transitory resource conflicts, these functions should be disabled at the *device* and not at a central controlling agent (for example, the interrupt controller). Devices that do not share any resources may have these resources disabled at a system level (for example, keyboard interrupts may be disabled at the interrupt controller in standard configurations).

Requirements:

2–1. I/O devices must adhere to the reset states given in Table 1 on page 10 when control of the system is passed from firmware to an operating system.

Table 1. I/O Device Reset States

Bus	Devices Left Open by Open Firmware	Other Devices
PCI	Interrupts not active No outstanding I/O operations Device is configured	The device is *inactive*: •I/O access response disabled •Memory access response disabled •PCI master access disabled •Interrupts not active Device is reset (see note)
ISA	Configured per NVRAM •Tri-state interrupts (inactive) •Tri-state DMA (inactive)	Hardware reset state (see note) •Tri-state interrupts (inactive) •Tri-state DMA (inactive)
PCMCIA	Configured per OF device tree	Powered off (see note)

Note: May optionally be *configured*, but must be *inactive*.

2–2. Prior to passing control to the operating system, firmware or hardware must initialize all registers not visible to the operating system to a state

Personal Use Copy - Not for Reproduction

that is consistent with the system view represented by the OF device tree.

2–3. Hardware must provide a mechanism, callable by software, to hard reset all processors and I/O subsystems in order to facilitate the implementation of the RTAS *system-reboot* function.

2–4. **For the Power Management option:** Hardware must provide a software-controllable mechanism to reset the I/O subsystems without affecting the state of the processors or memory to facilitate implementation of the RTAS *hibernate* function.

Hardware Implementation Note: Requirement 2–4 provides RTAS with a means to reset I/O that can only be reset by hardware independent of the device.

2.1.3.4 Locate an OS Boot Image

The operating system boot image is located as described in the *PowerPC Microprocessor Common Hardware Reference Platform System binding to: IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware* [10]. A device and filename can be specified directly from the command interpreter (the *boot* command) or OF will locate the image through an automatic boot process controlled by configuration variables. Once a boot image is located, the device path is set in the device tree as the *bootpath* property of the */chosen* node.

If multi-boot (multiple bootable operating systems residing on the same hardware platform) is supported, a configuration variable instructs the firmware to display a multi-boot menu from which the OS and bootpath are selected.

2.1.3.5 Load the Boot Image into Memory

After locating the image, it is loaded into memory at the location given by a configuration variable or as specified by the OS load image format.

2.1.4 Transfer Phase

The image is prepared for execution by checking it against certain configuration variables; this may result in a reboot (for example, *little-endian?*).

Personal Use Copy - Not for Reproduction

Once the operating system gains control, it may use the CIS interface to learn about the hardware platform contents and configuration. The OS will generally build its own version of this configuration data and may discard the OF code and device tree in order to reclaim the space used by Open Firmware. A set of platform-specific functions are provided by Run-Time Abstraction Services (RTAS) which is instantiated by the OS invoking the *instantiate-rtas* method of the RTAS Open Firmware device tree node.

2.1.5 Run-Time

During run-time, the operating system has control of the system and will have RTAS instantiated to provide low-level hardware-specific functions.

2.1.6 Termination

Termination is the phase during which the operating system yields control of the system and may return control to the firmware depending on the nature of the terminating condition.

2.1.6.1 Power Off

If the user activates the system power switch, power may be removed from the hardware immediately (switch directly controls the power supply) or software may be given an opportunity to bring the system down in an orderly manner (power management control of the power switch).

If power is removed from the hardware immediately, the operating system will lose control of the system in an undetermined state. Any I/O underway will be involuntarily aborted and there is potential for data loss or system damage. A shut-down process prior to power removal is highly recommended and has become standard in most modern operating systems.

In most power managed systems, power switch activation is fielded as a power management interrupt and the operating system (through RTAS) is able to quiesce the system before removing power. The operating system may turn off system power using the RTAS *power-off* function.

2.1.6.2 Suspend

The suspend state is entered upon the occurrence of various events in certain power managed systems. During suspend, only the contents of main memory

Personal Use Copy - Not for Reproduction

are preserved. Since the processor and all non-essential I/O may be turned off, a reboot-like process is required to restore the system.

To enter a suspend state, the operating system (OS) must call the RTAS *suspend* function. When system operation resumes, the firmware returns to the caller of the *suspend* function.

2.1.6.3 Hibernate

The hibernation state is entered upon the occurrence of various events in certain power managed systems. During hibernation, the system is effectively powered down. Memory contents are stored (normally to disk) and a reboot is required to restore the system.

RTAS provides a *hibernate* function to assist operating systems with saving the storage image. When the system reboots, the OS determines that it is waking up from hibernation and takes the appropriate actions to restore the system.

2.1.6.4 Reboot

The operating system may cause the system to reset and reboot by calling the RTAS *system-reboot* function.

2.2 Firmware

Requirements:

2–5. Platforms must implement Open Firmware as defined in *PowerPC Microprocessor Common Hardware Reference Platform System binding to: IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware* [10].

2–6. Platforms must implement the Run-Time Abstraction Services (RTAS) as described in Chapter 7, “Run-Time Abstraction Services,” on page 91.

2–7. Operating systems must use Open Firmware and the RTAS functions to be compatible with all platforms.

Hardware Implementation Note: Legacy Firmware may be implemented in addition to the above requirements. Legacy Firmware is described in the *PowerPC Reference Platform Specification,* Version 1.1 [28] including any addenda on the PowerPC forum on CompuServe.

Personal Use Copy - Not for Reproduction

Legacy firmware may be required by some operating systems until new versions are released which interface to Open Firmware.

Software Implementation Note: Operating systems may interface directly with the hardware without calling the RTAS functions until new versions of those operating systems are released which interface to RTAS.

Software Implementation Note: In addition to providing the RTAS interface, operating systems may provide direct hardware interfaces. An operating system which utilizes a platform specific interface instead of an RTAS function is called a "platform aware" operating system.

2.3 Bi-Endian Support

In Little-Endian mode, the address of a word in memory is the address of the least significant byte (the "little" end) of the word. Increasing memory addresses will approach the most significant byte of the word. In Big-Endian systems, the address of a word in memory is the address of the most significant byte (the "big" end) of the word. Increasing memory addresses will approach the least significant byte of the word.

Operating systems written to work in either Little-Endian or Big-Endian mode can operate on Common Hardware Reference Platform implementations. Firmware will initialize the platform to either Little-Endian (MSR_{LE}=1) or Big-Endian (MSR_{LE}=0) mode at boot time, depending on the need of the operating system.

This subject is covered in more detail in tutorials on endianess in *The PowerPC Architecture* [1], in *How to Create Endian-Neutral Software for Portability* [3], and in Appendix C, "Bi-Endian Designs," on page 265.

Requirements:

2–8. Platforms must support operation in Big-Endian mode.

2–9. Platforms must support operation in Little-Endian mode.

2.4 64-Bit Addressing Support

A *32-bit-addressing platform* is designed to operate with a 32-bit real address space (4 GB) only. Such a platform may contain either of the two processor (32-bit or 64-bit) implementations defined by the PowerPC microprocessor architecture, subject to the requirement 4–8 that a 64-bit processor implementa-

Personal Use Copy - Not for Reproduction

tion also implement the 32-bit-compatibility bridge architecture. Given this last requirement, the only difference in programming model between a 32-bit and 64-bit processor in a 32-bit-addressing platform is in the format of page table entries. Operating systems are required to support both formats (reference requirement 4–7).

A *64-bit-addressing-capable platform* is defined as one capable of supporting memory configured above 4 GB (greater than 32 bits of real addressing). This means that all hardware elements in the topology down to the Host Bridges are capable of dealing with a real address range greater than 32 bits, and all Host Bridges are capable of providing a translation mechanism for 32-bit I/O bus addresses. The initial mode of operation (when the operating system receives control from firmware) is 32-bit execution mode, including Host Bridges. All of System Memory is configured and reported by OF. (Note that this architecture allows only System Memory above 4 GB in the real address space.)

A *32-bit operating system* will utilize memory in the range up to 4 GB, and will operate a 64-bit-addressing-capable platform as a 32-bit-addressing platform with a 64-bit processor in it, regardless of how much memory is present. Memory above 4 GB will be unusable.

A *64-bit-addressing-aware operating system* is an OS that can deal with a real address space larger than 4GB. It must handle the 64-bit processor page table format (required of all OSs), and must understand Host Bridge mechanisms and Host Bridge Open Firmware methods for supporting System Memory greater than 4 GB.

On a 64-bit-addressing-capable platform with no memory configured above 4 GB, it is expected (although not required) that a 64-bit-addressing-aware OS would not enable 64-bit addressing in Host Bridges, and would operate the platform as if it were a 32-bit-addressing platform with a 64-bit processor. On a 64-bit-addressing-capable platform with memory configured above 4 GB, it is expected that a 64-bit-addressing-aware OS would recognize and use the memory above 4 GB (implying the enablement of 64-bit addressing in the Host Bridge(s)).

Software Implementation Note: The 64-bit addressing elements of the configuration are defined in the OF device tree as follows:

1. The *64-bit* property of the *cpu* nodes. This property is used to define the presence of a 64-bit PowerPC processor and the associated page table format (independent of the real addressing capability of the entire platform).

2. The *64-bit-addressing* property of the Host Bridge node(s). This property is required to be present on all HB nodes when the memory configuration has memory above 4 GB. It is reported so that

Personal Use Copy - Not for Reproduction

64-bit-addressing-aware OSs may recognize a 64-bit-addressing-capable platform and may choose to enable 64-bit addressing in Host Bridges even when no memory is configured above 4 GB. Open Firmware supplies a method on 64-bit-addressing-capable platforms (*set-64-bit-addressing*) which an OS uses to enable HB addressing of the full real address space above 4 GB.

3. Memory being reported above 4 GB in the *memory* node(s). This implies that the platform is 64-bit-addressing-capable.

2.5 Minimum System Requirements

This section summarizes the minimum hardware and functionality required for Common Hardware Reference Platform compliance.

The term *portable* is used to describe that class of systems that is primarily battery powered and is easily carried by its user.

The term *personal* is used to describe that class of systems that is bound to a specific work area due to its size or power source, and whose use is generally restricted to a single direct user or a small set of users.

The term *server* is used to describe that class of systems that supports a multi-user environment, providing a particular service such as file storage, software repository, or remote processing capability.

Each of these classes have unique requirements due to the way they are used or what operating systems they generally employ and, for this reason, the requirements that follow are based on the type of system being developed.

Requirements:

2–10. Platforms must contain the minimum required components given in Table 2 on page 19.

2–11. Portable and personal CHRP operating systems must support all the following:

a. PS/2™ *and* ADB™ keyboard/mouse interfaces.

b. 16550-compatible *and* SCC serial ports.

c. SCSI *and* IDE hard disk interfaces.

Software Implementation Note: Server-targeted CHRP operating systems are not subject to requirement 2–11 and may limit this I/O support to the intended target platform. Server-targeted CHRP operating

Personal Use Copy - Not for Reproduction

systems may, however, run on appropriately configured portable and/or personal platforms.

Hardware Implementation Note: Platform providers should consult OS providers for specific requirements of individual server-targeted operating systems.

2.5.1 Table Description

Minimum requirements for Common Hardware Reference Platforms are summarized in Table 2 on page 19.

2.5.1.1 Subsystem

Major subsystems are listed. This enumeration does not preclude the addition of new subsystems or the enhancement of listed subsystems; however, these must meet the definition and requirements given in Section 2.6, “Options and Extensions,” on page 20.

2.5.1.2 Specification

This column lists the *specification* of subsystems in a standard platform.

2.5.1.3 Portable, Personal, Server

These columns categorize the specifications according to the following legend.
Legend:

- R: Required.
- R*: Entries categorized as R* denote specifications from which platform vendors may *choose* one or more to satisfy the requirement. All R* specifications are supported by portable and personal CHRP operating systems.
- O: Optional (refer to the definition of optional given in Section 2.6, “Options and Extensions,” on page 20).
- M: Required for systems designed to run Apple Mac OS.

Personal Use Copy - Not for Reproduction

The categorization of the subsystem is given in the first line of each table entry; each individual specification is categorized separately in subsequent lines. If a subsystem is listed as optional, there may be specifications that are required *if* the particular subsystem is provided. This would be represented by an "O" in the first row of the entry with "R"s listed for the required specifications.If a subsystem is listed as required, a list of optional specifications means that a choice may be made from the list to fulfill the requirement.

Hardware Implementation Note: Platform providers are not required to provide all the I/O mechanisms labeled with R*. However, some application security and licensing measures will depend on them and platform limitations may result from their omission.

Hardware and Software Implementation Note: Server-targeted CHRP operating systems are not required to support all of the I/O mechanisms labeled with R* and thus platform providers should consult OS providers for specific requirements of individual server-targeted CHRP operating systems.

2.5.1.4 Description

The *description* column gives additional information about the requirements. Two documents are referenced extensively:

1. [20]*PowerPC Microprocessor Hardware Reference Platform: I/O Device Reference.* Use this document for additional information on the annotated specifications.
2. [23] *Macintosh Technology in the Common Hardware Reference Platform.* Use this document for specific physical, timing, and electrical requirements of the annotated specifications.

Personal Use Copy - Not for Reproduction

Table 2. Summary of Minimum Platform Requirements

Subsystem	Specification	Portable	Personal	Server	Description
Processor	PowerPC microprocessor	R	R	R	Further defined in Chapter 4, “Processor and Memory,” on page 49. Contact OS vendor for specific OS dependencies.
Minimum System Memory	8 MB expandable to at least 32 MB	R	R	R	Operating systems have specific requirements; contact OS vendor. All operating systems will run with 32 MB installed.
OS ROM or Socket	4 MB	R	R	M	[23]
Firmware Storage	Sized as needed	R	R	R	Most firmware implementations will fit in 256 KB. The preferred implementation is FLASH memory and should be processor-writable.
Non-Volatile Memory	8 KB	R	R	R	8 KB is sufficient for a system with a single operating system installed. Multi-boot systems must evaluate the need for more space. Each operating system may require up to 1 KB. See Chapter 8, “Non-Volatile Memory,” on page 141 for more information on Non-Volatile Memory.
External Cache		O	O	O	An external cache is recommended for performance. See Section 4.2.4, “Cache Memory,” on page 65 for more information.
Hard Disk	Sized as needed SCSI IDE PC Card	R O O O	R O O O	R O O O	[20], [23] “Medialess” systems are not covered by this architecture.
Floppy	 3.5” 1.44 MB MFM Media sense Auto eject Manual eject	O R R R R	O R R R R	O R R M R	[20] Not required, but a means to attach for software installation must be provided. This may be through a provided connector or over a network. Media sense: Implementations must allow polling of the drive up to 100x per second to determine the presence of media in the drive. A method for manual ejection of floppies is required.
CD-ROM	 4X speed ISO9660 Multi-session	O R R R	O R R R	O R R R	[20] Not required, but a means to attach for software installation must be provided. This may be through a provided connector or over a network
Alphanumeric Input Device	 PS/2 Keyboard interface ADB Terminal	R R* R* O	R R* R* O	O O O O	[20], [23] Servers do not require a keyboard for normal operation, however a means to attach an A/N Input Device must be provided; an ASCII terminal is a example of such a device. Keyboards must be capable of generating at least 101 scan codes.
Pointing Device	 2 buttons PS/2 interface (see note 1) ADB (see note 1)	R R R* R*	R R R* R*	O O O O	[20], [23] If a platform includes a keyboard, it must also include a pointing device with the functionality of at least a 2-button mouse

Personal Use Copy - Not for Reproduction

Table 2. Summary of Minimum Platform Requirements ***(Continued)***

Subsystem	Specification	Portable	Personal	Server	Description
Audio	16bit Stereo, 22.05 and 44.1 KHz, full duplex.	R	R	O	Servers do not require high quality audio. The programming model for this basic audio capability is specified in [20].
Tone		R	R	R	Must be implemented with separate hardware from system audio; must be capable of concurrent operation with system audio. May share speaker with system audio. See Table 29 on page 120 for the RTAS interface for tone generation.
Graphics		R	R	O	Servers do not require graphics during normal operation and need not support a graphics subsystem.
	1024x768	O	R	O	Portables may provide screen resolution in accordance
	Bi-Endian	R	R	O	with current state-of-the-art LCD technology.
	640x480x8 LFB	R	R	O	
	VGA	O	O	O	Some operating environments will prefer platforms that provide hardware support of VGA for performance considerations.
Real Time Clock		R	R	R	The RTC must be non-volatile and run continuously with a resolution of at least one second. See Section 7.3.3, “Time of Day,” on page 106 for further information.
VIA		R	R	M	[23]
Serial Port		O	R	O	
	16550	O	R*	O	[20]
	SCC	O	R*	O	[23]
Parallel Port	P1284 +ECP Mode	O	R	O	[20]
Network		O	O	O	
Interrupt Controller	Open PIC	R	R	R	[20]
	8259 tree	R	R	R	8259 required for ISA compatibility.
Direct Memory Access (DMA)	ISA	R	R	R	[20] ISA DMA must be 82378ZB compatible.
Power Management	States as defined.	R	O	O	Refer to Chapter 11, “Power Management,” on page 185
Infrared	IrDA	O	O	O	

2.6 Options and Extensions

An *option* is a system resource that is covered by this architecture, but is not required to be implemented. Platforms that implement options are required to conform to the definitions in this architecture, so that an aware OS environment can recognize and support them.

An *extension* is a resource that is outside the scope of, but does not contradict, this architecture. It is a resource which cannot be depended on by software to be present on a platform; operating systems are not required to be aware of, or capable of supporting, these resources.

Personal Use Copy - Not for Reproduction

Options and extensions must be dormant or invisible in the presence of a non-aware OS environment. In general this means they are initialized to an inactive state, and can be activated by an aware OS.

Requirements:

2–12. An option, if implemented, must operate as specified in this architecture.

2–13. Extensions, if implemented, must come up passively, such that an operating system which does not use the extension will not be affected.

2–14. Options, if implemented, must come up passively or as otherwise specified in this architecture.

2–15. An extension, if implemented, must not contradict this architecture.

Options provided by this architecture are listed below. Open Firmware configuration variable and properties associated with these options are given in *PowerPC Microprocessor Common Hardware Reference Platform System binding to: IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware* [10].

1. Power Management
2. Symmetrical Multiprocessing
3. 64 bit support
4. PC Emulation
5. Multi-boot

Personal Use Copy - Not for Reproduction

Blank page when cut - 22

System Address Map

3

The address map of a Common Hardware Reference Platform is made up of several distinct areas. These areas are one of five basic types. Each of these types has its own general characteristics such as coherency, alignment, size restrictions, variability of starting address and size, the system action on access of the area, and so on. This chapter will give details on some of those characteristics, and other chapters will define the other characteristics. The variable characteristics of these areas will be reported to the operating system via properties in the OF device tree.

3.1 Address Areas

The following is a definition of the five areas and some of their characteristics:

- *System Memory* refers to memory which forms a coherency domain with respect to the PowerPC processor(s) that execute application software on a system. See Section 4.2.2.1, “Memory Coherence,” on page 57 for details on aspects of coherence. *System Memory Spaces* refer to one or more pieces that together form the System Memory.
- *Peripheral Memory Space* refers to a range of real addresses which are assigned to the Memory Space of a Host Bridge (HB) and which are sufficient to contain all of the *Load* and *Store* address space requirements of the devices in the Memory Space of the I/O bus that is generated by the HB. The frame buffer of a graphics adapter is an example of a device which may reside in the Peripheral Memory Space. In addition to a Memory Space, many

types of I/O buses have a separate address space called the I/O Space. An HB which generates such I/O buses must decode another address range, the Peripheral I/O Space.[1]

- *Peripheral I/O Space* refers to a range of real addresses which are assigned to the I/O Space of an HB and which are sufficient to contain all of the *Load* and *Store* address space requirements of all the devices in the I/O Space of the I/O bus that is generated by the HB, including the expansion of those addresses due to the discontiguous I/O address mode. A keyboard controller is an example of a device which may require Peripheral I/O Space addresses.
- *System Control Area* refers to a range of addresses which contains the system ROM(s) (firmware ROM and, if implemented, OS-ROM) and an unarchitected, reserved, platform-dependent area used by firmware and Run-Time Abstraction Services for control of the platform. The ROM areas are defined by the OF properties in the *openprom* and *os-rom* nodes of the OF device tree. The Mac OS-ROM definition, for implementations which require it, can be found in *Macintosh Technology in the Common Hardware Reference Platform* [23].
- *Undefined* refers to areas that are not one of the above four areas. The result of accessing one of these areas is defined in Chapter 10, "Error and Event Notification," on page 157 as an "invalid address" error.

In addition to these five major areas, there are several subsets of these areas for the first HB (HB0) of a platform. Note that HB0 must be a PCI Host Bridge (PHB) so HB0 will be indicated by PHB0 from here on. These subsets may be optional and may modify the general characteristics of the area. They include:

- *Compatibility holes* are address decodes on PHB0 of a platform which are defined for use by software which needs compatibility with existing PC systems relative to the requirement for specific holes in the System Memory and I/O bus Memory Spaces. These holes may also impact the design of the System Memory controller. These holes can be enabled independently, but the platform need not provide the capability to enable the processor-hole while the io-hole is disabled.
  - *io-hole* is an address range on the I/O bus side of PHB0. Platforms which provide support for Video Graphics Array (VGA) compatibility will support the io-hole. When enabled, this creates a hole in the first System Memory Space (as viewed by the I/O) from 640 KB to (1 MB - 1), and

[1] A peripheral space may also include a "configuration" address space. The configuration space will be abstracted by a Run-Time Abstraction Service (for example, see Section 7.3.5, "PCI Configuration Space," on page 114).

Personal Use Copy - Not for Reproduction

accesses by the I/O to this address range will remain on the I/O bus (allowing I/O device to device transfers in this address range). The existence of this hole is indicated by the existence of the *io-hole* node property in the PHB0 node of the OF device tree. The hole is activated or deactivated by the *set-io-hole* OF method.

— *processor-hole* is an optional address range on the processor side of PHB0. When enabled, this creates a hole in the first System Memory Space (as viewed by the processor) from 640 KB to (768 KB - 1), and causes accesses in this address space to go to the Memory Space of the I/O bus of PHB0, in the 640 KB to (768 KB - 1) range (thus giving the Peripheral Memory Space characteristics to this address range in System Memory). The existence of this optional hole is indicated by the existence of the *processor-hole* node property in the PHB0 node of the OF device tree. The hole is activated or deactivated by the *set-processor-hole* OF method.

■ *Initial memory alias spaces* are areas in the Peripheral Memory Space of PHB0 which allow accessing the first (initial) 16 MB address range (0 to (16 MB - 1)) on one side of PHB0 from a different address range on the other side of PHB0. These spaces are required to be enabled when either of the compatibility holes are enabled. It is necessary to enable these alias spaces on PHB0 in order to support devices which must be configured in the 0 to (16 MB - 1) range. The peripheral-memory-alias and system-memory-alias spaces are enabled or disabled as a pair, with one exception, via the *set-initial-aliases* OF method for PHB0. The exception is when the PC Emulation option is enabled (see Section 3.3, “PC Emulation Option,” on page 44). The state of the alias spaces (enabled or disabled) is indicated by the *initial-memory-alias* property of PHB0 node. The initial memory alias spaces are entirely contained within the Peripheral Memory Space.

— *peripheral-memory-alias space* is an address space on the processor side of PHB0. When enabled, this address range allocates the 16 MB of address space at the high end of the Peripheral Memory Space for PHB0, and this is used to address the first 16 MB of the Memory Space on the I/O side of PHB0. This address range is used to access I/O devices which cannot have their addresses set in the Peripheral Memory Space. For example, ISA devices or ISA-compatible devices must be setup to have addresses below 16 MB.

— *system-memory-alias space* is an address space on the I/O side of PHB0. When enabled, this address range, which is at the same address in the I/O bus Memory Space as the peripheral-memory-alias on the processor side, can be used to address the first 16 MB of System Memory. This ad-

Personal Use Copy - Not for Reproduction

dress range is used, for example, to access System Memory in the 640 KB to (1 MB - 1) range when the io-hole is enabled (see also Table 10 on page 79 for other uses).

Table 3 on page 26 shows the valid combinations of these alias and hole areas for normal system operations. Of these combinations, the rows which indicate the valid states when the OF passes control to the operating system are States 1 and 2, depending on whether the initial memory alias spaces are enabled or not.

Table 3. Valid Hardware Combinations of Compatibility Holes and Initial Memory Alias Spaces

State Number	PC Emulation Option	Peripheral-memory-alias	System-memory-alias	Processor-hole	Io-hole
1	disabled	disabled	disabled	disabled	disabled
2	disabled	enabled	enabled	disabled	disabled
3	disabled	enabled	enabled	disabled	enabled
4	disabled	enabled	enabled	enabled	enabled
5	enabled	enabled	disabled	disabled	enabled
6	enabled	enabled	disabled	enabled	enabled

Software Implementation Note: Software should be careful to assure that no I/O operations using the alias spaces or holes are occurring during the transition period from one state in the table to another. Software may pass through states which do not appear in the table during the transition from one state in the table to another.

In describing the characteristics of these various areas, it will be convenient to have a nomenclature for the various boundary addresses. Table 4 on page 26 defines the labels which will be used when describing the various address ranges. Note that “bottom” refers to the smallest address of the range and “top” refers to the largest address.

Table 4. Map Legend

Label	Description
BIOn	Bottom of Peripheral I/O Space for HBn (n=0, 1, 2,...). The OF property *ranges* in the device tree for HBn will contain the value of BIOn.

Personal Use Copy - Not for Reproduction

Table 4. Map Legend ***(Continued)***

Label	Description
TIOn	Top of Peripheral I/O Space for HBn (n=0, 1, 2,...). The value of TIOn can be determined by adding the size of the area as found in the OF property *ranges* in the OF device tree for HBn to the value of BIOn found in that same property.
BPMn	Bottom of Peripheral Memory Space for HBn (n=0, 1, 2,...). The OF property *ranges* in the device tree for HBn will contain the value of BPMn.
TPMn	Top of Peripheral Memory Space for HBn (n=0, 1, 2,...). For PHB0, when the initial memory alias spaces are enabled, the Peripheral Memory Space is split into two pieces in the OF device tree, as indicated by the *ranges* property in the OF device tree for the PHB0 node; BPM0 to (BIM - 1) and BIM to TPM0. When the initial memory alias spaces are disabled or for other HBs other than PHB0, the Peripheral Memory Space is in one piece in the OF device tree, as indicated by the *ranges* property in the OF device tree for HBn node; BPMn to TPMn. The value of TPMn can be determined by either adding the size of the area as found in the OF property *ranges* in the OF device tree for HBn to the value of BPMn found in that same property or by adding the size of the area as found in the OF property *ranges* in the OF device tree for HBn to the value of BIM found in that same property, depending on whether the initial memory alias spaces are disabled or enabled, respectively.
BIM	Bottom of initial memory alias space for PHB0. Corresponding top of initial memory alias space is equal to TPM0. The value of BIM is (TPM0 - 16 MB + 1). If the initial memory alias spaces are enabled, then the value of BIM is indicated by the *ranges* property in the OF device tree for PHB0 node, otherwise it can be calculated by (TPM0 - 16 MB + 1) (see explanation under TPMn, above).
BSCA	Bottom of System Control Area. Corresponding top of the System Control Area is (4 GB - 1). The OF property *reg* in the OF device tree for the System Control Area node contains the value of BSCA.
BSMn	Bottom of System Memory Space n (n=0, 1, 2,...); BSM0 = 0. The OF property *reg* in the OF device tree for the Memory Controller node contains the value of BSMn.
TSMn	Top of System Memory Space n (n=0, 1, 2,...). The value of TSMn can be determined by adding the value of BSMn as found in the Memory Controller node of the OF device tree to the value of the size of that area, found in the same property.

Operating systems and other software should not use fixed addresses for these various areas. A given platform may, however, make certain of these addresses unchangeable. Each of these areas will be defined in the OF device tree in the node of the appropriate controller. This will give platforms maximum flexibility in implementing the System Address Map to meet their market requirements. Some of the values set up for these areas by the OF may be changeable by an OF method. Refer to the chapters describing the various controller architectures for more information.

Personal Use Copy - Not for Reproduction

Requirements:

3–1. All unavailable addresses in the Peripheral Memory and Peripheral I/O Spaces must be conveyed in the OF device tree.

a. A device type of *reserved* must be used to specify areas which are not to be used by software and not otherwise reported by OF.

b. Shadow aliases must be communicated as specified by the appropriate OF bus binding.

3–2. There must not be any address generated by the system which causes the system to hang.

Hardware Implementation Note: The reason for requirement 3–1 is to reserve address space for registers used only by the firmware or addresses which are used only by the hardware.

3.2 Address Decoding and Translation

In general, different components in the hardware are going to decode the address ranges for the various areas. In some cases the component may be required to translate the address to a new address as it passes through the component. The requirements, below, describe the various system address decodes and, where appropriate, what address transforms take place outside of the processor. The details are clarified by the example system address maps which can be found in Figure 4 on page 33, Figure 5 on page 34, and Figure 6 on page 35. Note that Direct Memory Access (DMA) operations can either be controlled directly by the device doing the I/O operation (that is, an I/O bus master operation), or may be controlled by a controller which is separate from the device (that is, a third party DMA operation).

The HB requirements in this section refer to HBs which are defined by the CHRP architecture. Currently, there is only one HB defined by the CHRP architecture, and that is the PHB. HBs which implement I/O buses other than those defined by the CHRP architecture are encouraged to use this addressing model. However, HBs which are implemented as extensions, are in the disabled (passive) state when control is passed to the operating system, and therefore need not implement the same programming model.

Requirements:

3–3. Processor *Load* and *Store* operations must be routed and translated as shown in Table 5 on page 32.

Personal Use Copy - Not for Reproduction

3–4. DMA operations in the Memory Space of an I/O bus must be routed and translated as shown in Table 6 on page 32.

3–5. In addition to Table 6 on page 32, if the platform is designed to support System Memory configured at an address of 4 GB or above, then the Translation Control Entry (TCE) translation mechanism, described in Section 3.2.2, "Translation of 32-Bit DMA Addresses in 64-Bit Addressing Systems," on page 38 must be implemented on all HBs. If the operating system enables the platform to access System Memory at or above 4 GB, then TCEs must be used to translate all DMA operations in the Memory Space of the I/O bus of the HB which use a 32-bit address.

3–6. An HB must not act as a target for operations in the I/O Space of an I/O bus.

3–7. The following are the System Control Area requirements:

a. Each platform must have exactly one System Control Area.

b. The System Control Area must not overlap with the System Memory Space(s), Peripheral Memory Space(s), or the Peripheral I/O Space(s) in the platform.

3–8. The following are the System Memory Space requirements:

a. Each platform must have at least one System Memory Space.

b. The System Memory Space(s) must not overlap with the Peripheral I/O Space(s), Peripheral Memory Space(s), the System Control Area, or other System Memory Space(s) in the platform.

c. The first System Memory Space must start at address 0 (BSM0 = 0), must be at least 16 MB before a second System Memory Space is added, and must be contiguous except that if the processor-hole is enabled, then there will be a hole from 640 KB to (768 KB - 1).

d. Each of the additional (optional) System Memory Space(s) must start on a 4 KB boundary.

e. Each of the additional (optional) System Memory Space(s) must be contiguous within itself.

f. There must be at most eight System Memory Spaces below BSCA and at most eight at or above 4 GB.

Personal Use Copy - Not for Reproduction

g. If multiple System Memory Spaces exist below BSCA, then they must not have any Peripheral Memory or Peripheral I/O Spaces interspersed between them.

3–9. The following are the Peripheral Memory Space requirements:

a. The Peripheral Memory Space(s) must not overlap with the System Memory Space(s), Peripheral I/O Space(s), the System Control Area, or other Peripheral Memory Space(s) in the platform.

b. When the OF passes control to the operating system, there must be no I/O device configured in the address range (TPM0 - 16 MB + 1) to TPM0 in the Peripheral Memory Space for PHB0, in order to assure that 16 MB of space is available for the initial memory alias spaces.

c. The size of each Peripheral Memory Space (TPMn - BPMn + 1) must be a power of two for sizes up to and including 256 MB, with the minimum size being 16 MB, and an integer multiple of 256 MB plus a power of two which is greater than or equal to 16 MB for sizes greater than 256 MB (for example, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 256 + 16 MB, 256 + 32 MB,..., 512 + 16 MB,...).

d. The boundary alignment for each Peripheral Memory Space must be an integer multiple of the size of the space up to and including 256 MB and must be an integer multiple of 256 MB for sizes greater than 256 MB.

e. There must be exactly one Peripheral Memory Space per HB.

f. The Peripheral Memory Space for every HB defined by the CHRP architecture must reside below BSCA.

3–10. The following are the Peripheral I/O Space requirements:

a. The Peripheral I/O Space(s) must not overlap with the System Memory Space(s), Peripheral Memory Space(s), the System Control Area, or other Peripheral I/O Space(s) in the platform.

b. The size of each Peripheral I/O Space (TIOn - BIOn + 1) must be a power of two with the minimum size being 8 MB (that is, sizes of 8 MB, 16 MB, 32 MB, 64 MB, and so on, are acceptable).

c. The boundary alignment for each Peripheral I/O Spaces must be an integer multiple of the size of the space.

d. There must be at most one Peripheral I/O Space per HB.

Personal Use Copy - Not for Reproduction

e. Peripheral I/O Spaces for all HBs defined by the CHRP architecture must reside below BSCA.

3–11. The following are the initial memory alias space requirements:

a. The peripheral-memory-alias and system-memory-alias spaces must be implemented on PHB0, and these initial memory alias spaces must have the capability to be enabled or disabled by the *set-initial-aliases* OF method for PHB0 node.

b. The initial state of the peripheral-memory-alias and system-memory-alias spaces, when control passes from OF to the operating system, must be enabled if there is any device on the I/O side of PHB0 which is configured in the 0 to (16 MB - 1) address range of the Memory Space of the I/O bus, and must be disabled otherwise.

3–12. The following are the compatibility hole requirements:

a. The initial state of the io-hole and the processor-hole, when control passes from OF to the operating system, must be disabled (if implemented).

b. If a platform implements the PC Emulation option, then the io-hole must be implemented. See Section 3.3, “PC Emulation Option,” on page 44 for more information.

c. Platforms for which VGA support is provided or which have an ISA bus must also implement the io-hole (see Table 2 on page 19 for the platform VGA requirements).

3–13. I/O devices which cannot be configured in the Peripheral Memory Space address range must be located on the I/O bus of PHB0, or on another I/O bus which is generated by a bridge attached to this bus, and must be configured in the 0 to (16 MB - 1) address range.

Hardware Implementation Note: OF may wish to display an error message to the user if it finds a device which cannot be configured due to requirement 3–13.

After a DMA accesses passes through an HB it is routed and translated as per Table 5 on page 32. Platforms do not need to support I/O device DMA access to the ROM areas.

The figures beginning with Figure 4 on page 33 show examples of system address maps.

Personal Use Copy - Not for Reproduction

Table 5. Processor Bus Address Space Decoding and Translation

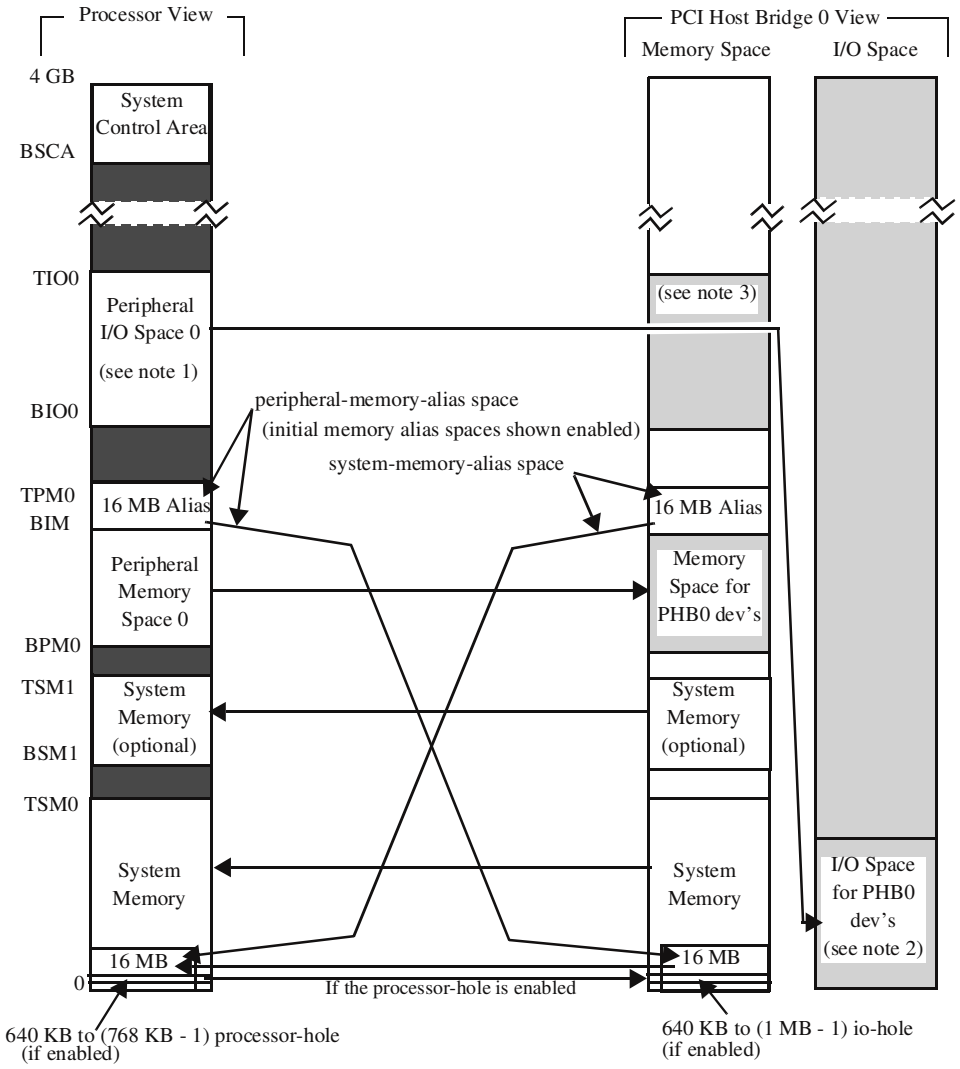
Address Range at Processor Bus	Route and Translation Requirements	Other Requirements and Comments
BSCA to (4 GB - 1)	To ROM controller or to a platform dependent area. Translation dependent on implementation.	Areas other than ROM are reserved for firmware use, or will have their address passed by the OF device tree.
BIOn to TIOn (n=0,1,...)	HBn responds. For requirements, see Section 3.2.1, “Peripheral I/O Address Translation,” on page 37.	
BPMn to TPMn (n=0, 1,...)	Send through HBn to the Memory Space of the I/O bus. For PHB0, if the peripheral-memory-alias is enabled, then translate an address in the peripheral-memory-alias space (BIM to TPM0; 16 MB range) so that this address range becomes 0 to (16 MB - 1), otherwise do not translate.	
BSMm to TSMm (m>0)	To System Memory Space m, no translation.	Can be at or above 4 GB, or below BSCA.
0 to TSM0	No translation. If the processor-hole is disabled or not implemented, then send entire range to System Memory. If processor-hole is enabled, then send 640 KB to (768 KB - 1) through PHB0 to the Memory Space of the I/O bus and send the remainder of the range to System Memory.	
All other addresses	See Chapter 10, “Error and Event Notification,” on page 157.	Access is to undefined space.

Table 6. DMA Address Decoding and Translation (I/O Bus Memory Space)

Address Range at I/O Side of HBn	Route and Translation Requirements	Other Requirements and Comments
BIOn to TIOn (note 1)	HBn either does not respond or responds and signals an error to the device (see Chapter 10, “Error and Event Notification,” on page 157).	This is an error.
BPMn to TPMn (note 1)	For PHB0, if the system-memory-alias space is enabled and the access is to the system-memory-alias space (BIM to TPM0), then translate so that this address range becomes 0 to (16 MB - 1) and send through PHB0 (note 2), otherwise the HBn does not respond.	If not an access to the system-memory-alias space, this is an I/O device to device transfer.
640 KB to 1 MB - 1 (PHB0 only)	If the io-hole is disabled or not implemented, then pass through PHB0 untranslated (note 2). If the io-hole is enabled, PHB0 does not respond.	
BSCA to (4 GB - 1)	May pass through HB untranslated, or HB may not respond.	Implementation dependent.
All other Memory Space addresses	Pass through HB. If the address is greater than or equal to 4 GB, see requirements 3–22 and 3–23 for translation requirement. If the address is less than 4 GB, then do not translate unless note 2 is applicable.	Valid accesses are to System Memory or possibly to other HB Peripheral Memory Spaces.

Note 1: n = # of HB Viewing or Receiving the Operation.
Note 2: If the HB 64-bit-addressing option is enabled, translate via the TCE before passing through the HB.

Personal Use Copy - Not for Reproduction



Note 1: BIOn + 64 KB to BIOn + (8 MB - 1) are not accessible when the discontiguous mode is disabled.

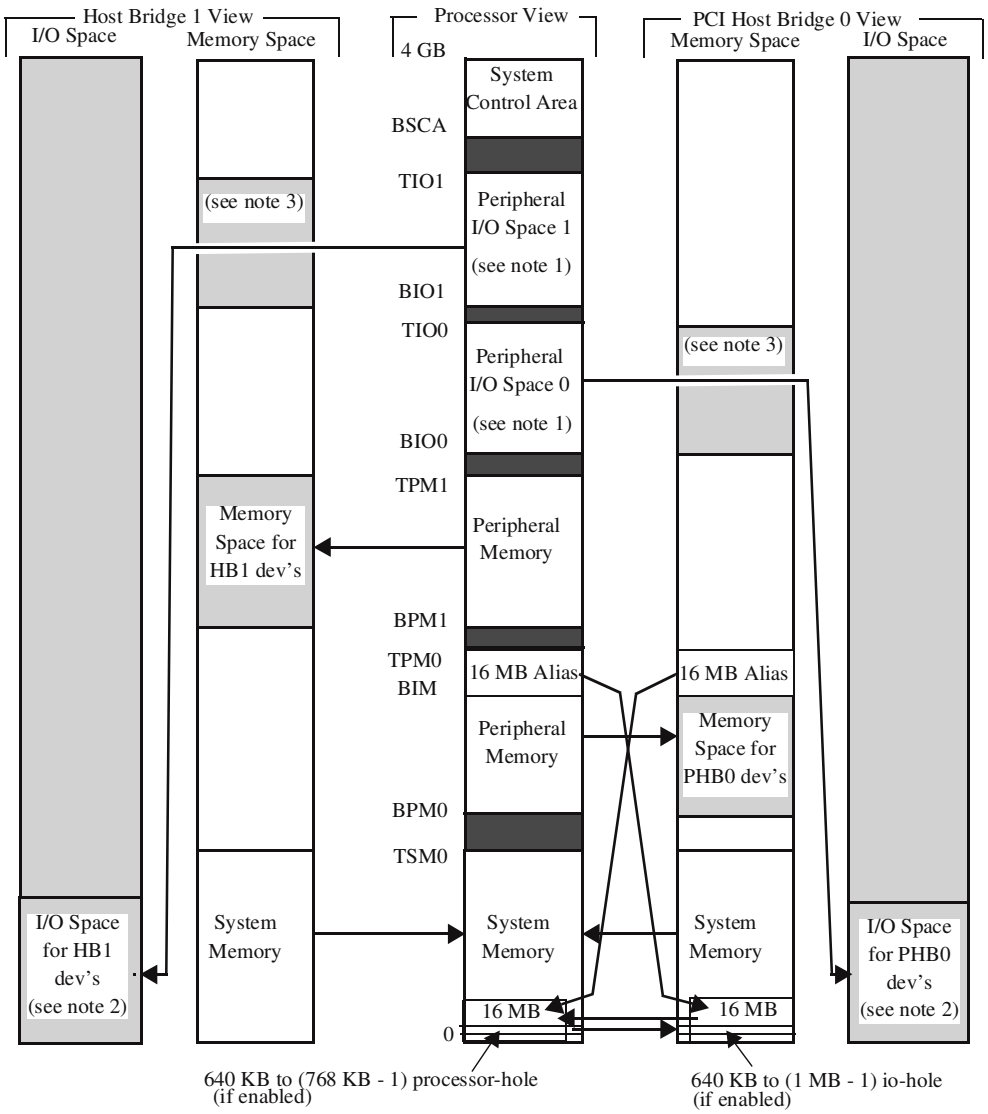
Note 2: Addresses from 64 KB to (8 MB - 1) are not accessible.

Note 3: HB may optionally respond to addresses from BIOn to TIOn and signal an error to the device.

= Addresses which are invalid for *Load* and *Store* = PHB does not respond

Figure 4. Example of an Address Map for a 32-Bit Addressing System with One PHB

Personal Use Copy - Not for Reproduction



Notes 1, 2, and 3 refer to the same notes as in Figure 4 on page 33.

= Addresses which are invalid for *Load* and *Store* = HB does not respond

Figure 5. Example of an Address Map for a 32-Bit Addressing System with Two HBs

Personal Use Copy - Not for Reproduction

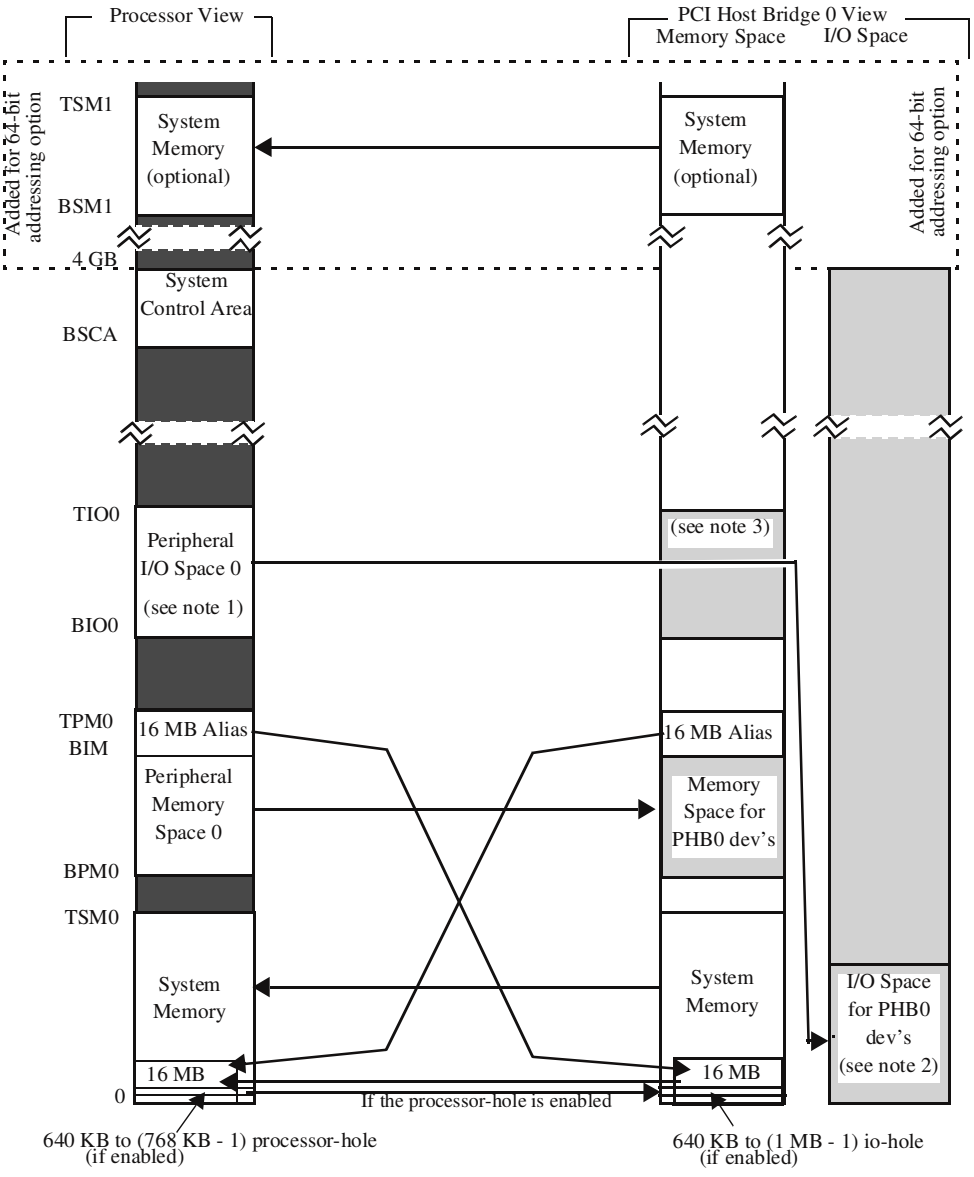


Figure 6. Example of an Address Map for a 64-Bit Addressing System with One PHB

Personal Use Copy - Not for Reproduction

The beginning addresses and sizes of the Peripheral I/O Space(s) and Peripheral Memory Space(s), as set up by OF, may have values which are fixed by the platform or may be changeable by the *change-address-map* OF method. The *change-address-map* OF method will return a failure indication if it was not able to change the map (for example, if the address area is unchangeable in the platform, or if an invalid or unsupported size or alignment was specified). The OF node property corresponding to the boundary and size of these areas is the *ranges* property of the HB node. When a platform allows these areas to be changed, they can be changed to a size and alignment according to requirements 3–9c, 3–9d, 3–10b, and 3–10c.

Hardware and Software Implementation Notes:

1. The *change-address-map* OF method does not need to support all possible sizes for the areas to which it allows changing; it is permissible to return a failure indication on some combination of sizes and alignments allowed by requirements 3–9 and 3–10, but to allow others.

2. If software receives a failure indication from *change-address-map* it may try other valid size and alignment combinations, in case the platform does not support all valid size and alignments but does support some (see note 1, above).

3. A platform which supports changing of the address map via the *change-address-map* OF method should support all sizes via that method which can be reported by OF during boot time. This allows the operating system to set the map after a wakeup back to what it was before a hibernate, in the case that the configuration and address map have changed during the time between hibernate and wakeup.

Support for the io-hole may also be needed to support certain peer to peer operations (see Table 10 on page 79).

Certain System Memory addresses must be reserved in all systems for specific uses (see Section 4.1.2, “PowerPC Microprocessor Differences,” on page 51 for more information).

Before leaving this section, it is important to note (because it is somewhat buried in various requirements, above) that all I/O devices which cannot be configured at an address in the Peripheral Spaces of a HB (for example, VGA or ISA devices which have to be configured below 16 MB), must reside in the Peripheral Space of PHB0 and be on the PCI bus generated by PHB0 or on an ISA bus or another I/O bus which is generated from that PCI bus.

Personal Use Copy - Not for Reproduction

3.2.1 Peripheral I/O Address Translation

The CHRP architecture defines a discontiguous I/O address mode which allows for the expansion of low 64 KB of the I/O Space of the I/O bus to an 8 MB address space in the processor's Peripheral I/O Space. This mode is enabled by the *set-discontiguous-io* OF method. When enabled, this expansion puts each 32 bytes of the lowest 64 KB of the I/O Space of the I/O bus in its own 4 KB page in the processor's Peripheral I/O Space. Enabling this mode allows operating systems to add protection for accesses to an I/O Space of the I/O bus on a 32 byte granularity (using the normal processor page protection mechanisms). This is illustrated in Figure 7 on page 38.

Requirements:

3–14. When the discontiguous I/O mode is enabled, Processor *Load* and *Store* addresses in the first 8 MB of Peripheral I/O Space experience the following translation. The high order seven bits of each 4 KB page offset are ignored. Thus, all page offsets wrap to the same 32 bytes within that page. Successive page numbers, starting at BIO, reference successive 32-byte blocks of Peripheral I/O Space, starting at address 0 (see Figure 7 on page 38).

3–15. When the discontiguous I/O mode is disabled (that is, in the contiguous mode), addresses are translated so that BIO to (BIO + 64 KB - 1) is sent through to the I/O Space of the I/O bus starting at address 0, and (BIO + 8 MB) to TIO is sent through to the I/O Space of the I/O bus starting at 8 MB (see Figure 7 on page 38).

3–16. The discontiguous I/O mode must be disabled when control is passed to the operating system.

Addresses in the Peripheral I/O Space from BIO + 64 KB to BIO + (8 MB - 1) must be reserved for hardware use.Addresses in the Peripheral I/O Space from (BIO + 64 KB) to (BIO + (8 MB - 1)) are reserved for hardware use. HBs may treat the areas as reserved or may pass them through to the I/O Space of the I/O bus at addresses 64 KB to (8 MB - 1). I/O devices should not be configured in the 64 KB to (8 MB - 1) range for two reasons. First, hardware may not pass this address range through the HB when the discontiguous mode is disabled. Secondly, if the platform is switched from contiguous to discontiguous mode, these devices would no longer be accessible if configured in this range.

Personal Use Copy - Not for Reproduction

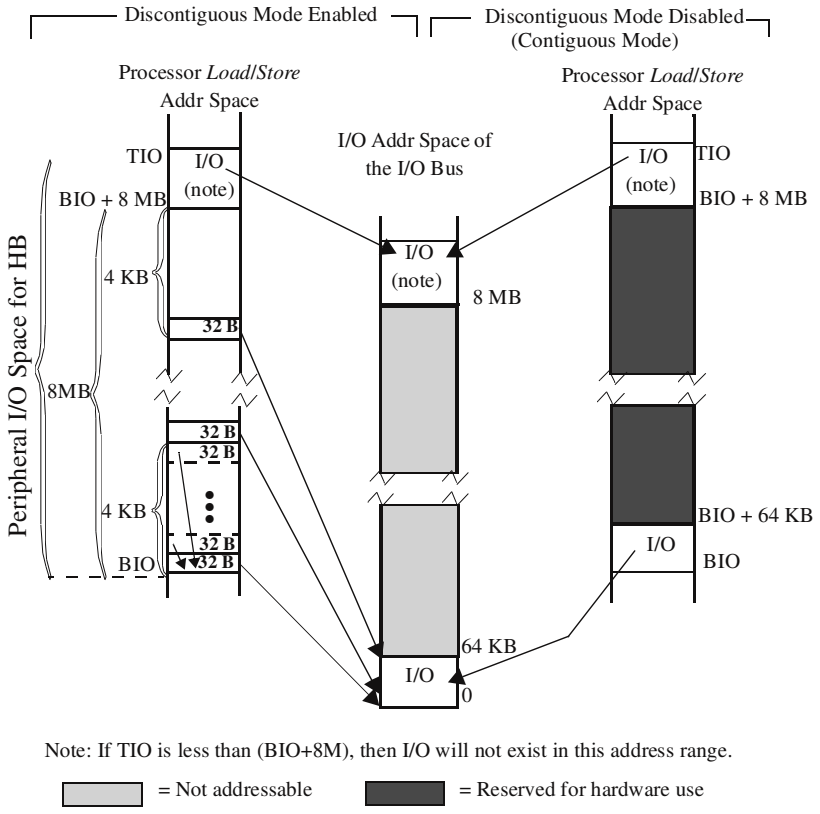


Figure 7. Models for *Load* and *Store* Instructions to Peripheral I/O Space

3.2.2 Translation of 32-Bit DMA Addresses in 64-Bit Addressing Systems

I/O devices which are only capable of accessing up to 4 GB via DMA need a way to access above that limit when used in 64-bit addressing systems and the addressing requirements go beyond 4 GB. This could be accomplished by the device driver transferring the data to a DMA buffer in System Memory in the first 4 GB of address space, and then transferring the data at or above 4 GB by doing a software move of the data. This, however, will not perform well, and it is in these large systems where performance is generally the most critical.

Personal Use Copy - Not for Reproduction

Therefore, the CHRP architecture defines a mechanism for translating DMA addresses for I/O devices which are only capable of addressing a 4 GB I/O Memory Space, to the larger address space required in some 64-bit addressing systems. This capability in an HB will be called the 64-bit addressing option, and is required to be implemented if the platform is designed to support System Memory configured at an address of 4 GB or above.

The 64-bit addressing option is disabled when the operating system is given control. After the operating system has determined that the platform contains System Memory at or above 4 GB, it may assume that all HBs support the 64-bit addressing option and enable the option by using the *set-64-bit-addressing* OF method in each HB node of the OF device tree (each HB has to be enabled separately in order to set up the TCE table address and size for each). The operating system can determine the existence of HB support of 64-bit addressing option by looking for the existence of the *64-bit-addressing* property in the HB node(s) of the OF device tree and if all HBs support the 64-bit addressing option, the operating system may enable the HBs' 64-bit addressing option, even if there is no System Memory configured at an address of 4 GB or above (that is, 64-bit capable systems can be enabled to support the Translation Control Entry (TCE) translation mechanism to translate DMA operations, even if they don't have any System Memory located at or above 4 GB).

Requirements:

3–17. For 64-bit addressing option in HBs: In platforms which are designed to support System Memory configured at an address of 4 GB or above, the TCE mechanism, described in Section 3.2.2, "Translation of 32-Bit DMA Addresses in 64-Bit Addressing Systems," on page 38, must be implemented on all HBs. After a DMA operation is accepted by the HB and pre-translated as per Table 6 on page 32, if the 64-bit addressing option of the HB is enabled, the HB must use the TCE mechanism to translate the address when the I/O device presents a 32-bit address to the Memory Space of the HB.

3–18. For 64-bit addressing option in HBs: The bits of the TCE must be implemented as defined in Table 5 on page 32.

3–19. For 64-bit addressing option in HBs: Enough bits must be implemented in the TCE so that I/O DMA devices are able to access all System Memory addresses.

3–20. For 64-bit addressing option in HBs: TCEs must be stored as Big-Endian entities.

Personal Use Copy - Not for Reproduction

3–21. For 64-bit addressing option in HBs: When the 64-bit addressing option is enabled, an HB must not accept 32-bit accesses unless they would also be accepted under the requirements in Table 6 on page 32.

3–22. For 64-bit addressing option in HBs: If an HB accepts 64-bit addresses on DMA accesses (as reported by the existence of the *64-bit-dma* property in the HB node of the OF device tree for that specific HB) and if the 64-bit addressing option of the HB is enabled, then that HB must not use TCEs to translate I/O bus Memory Space DMA addresses which support a full 64-bit address (for example, Dual Address Cycle (DAC) accesses on the PCI bus do not use the TCE translation mechanism in the PCI HB (PHB)).

3–23. For 64-bit addressing option in HBs: If an HB accepts 64-bit addresses on DMA accesses and if the 64-bit addressing option of the HB is enabled, that HB must translate 64-bit I/O bus Memory Space DMA accesses (for example, DAC PCI accesses for the PHB) in the upper 4 GB of the 64-bit Memory Space of the I/O bus to the 0 to (4 GB - 1) address range before passing the access to the host side of that HB (for example, for a PHB, if AD[63:32] (PCI notation) are equal to 0xFFFF FFFF, then the PHB must set AD[63:32] to 0), otherwise the HB must not translate 64-bit accesses (see Figure 8 on page 43).

3–24. For 64-bit addressing option in HBs: After translation of the address via requirements 3–17 or 3–23, above, an HB must use the translated address to access the system, unless that address would re-access the same HB (for example, is in the Peripheral Memory Space or Peripheral I/O Space of that HB), in which case the HB should generate an invalid address error. (See Chapter 10, “Error and Event Notification,” on page 157)

3–25. For 64-bit addressing option in HBs: TCEs must be located in System Memory or appear to software as though they are in System Memory, the memory must be a contiguous real address range, and the memory must be coherent.

3–26. For 64-bit addressing option in HBs: Each HB must provide the capability of having its own independent TCE table.

3–27. For 64-bit addressing option in HBs: Any non-recoverable error while an HB is accessing its TCE table must result in a TCE access error; the action to be taken by the HB being defined under the TCE access error in Chapter 10, “Error and Event Notification,” on page 157.

Personal Use Copy - Not for Reproduction

3–28. For 64-bit addressing option in HBs: In implementations which cache TCEs, if software issues a *Store* instruction to a TCE, then the hardware must perform the following steps: First, if any data associated with the page represented by that TCE is in an I/O bridge cache or buffer, the hardware must write the data, if modified, to System Memory. Secondly, it must invalidate the data in the cache. Finally, it must invalidate the TCE in the cache.

3–29. For 64-bit addressing option in HBs: Neither an I/O device nor an HB must ever modify a TCE.

3–30. For 64-bit addressing option in HBs: If the page mapping and control bits in the TCE are set to 0b00, the hardware must not change its state based on the values of the remaining bits of the TCE.

Software Implementation Note: Software must be careful when attempting DMA operations through an HB to that HB's own TCE table, because HB implementations which cache TCEs are not required to detect changes to the TCEs that they have cached while doing a DMA operation to their own TCE table.

The translation in requirement 3–23 is so that an I/O device which is capable of accessing a full 64-bit address space can get access to the first 4 GB of the address space without going through a TCE translation. Normally, one would expect that this would be possible by the 64-bit device just issuing a DMA operation with an address of less than 4 GB. However, some I/O buses (for example, PCI) do not have a protocol where the I/O device tells the HB that they are capable of greater than 32 bits of addressing when the address is less than 4 GB. In such a case, the HB will not be able to tell the difference between a 32-bit device and a 64-bit device for addresses below 4 GB, and therefore has to assume a 32-bit device and use TCEs. Note that none of this precludes having 32-bit and 64-bit I/O devices on the same bus at the same time.

Figure 8 on page 43 is a representation of how the 32-bit to 64-bit TCE and 64-bit to 64-bit translations work. The information for the translation from 32 bits to the number of address bits required by the platform is contained in the TCE entries in a TCE table of an HB. The I/O bus address is first checked to see if the access is targeted to the address space on the other side of the HB. Since there is not a way for the device to indicate that the operation is headed to the other side of the HB, the selection process is by the same method as for the 32-bit addressing system case; that is, an HB will accept a 32-bit address for translation via a TCE if the access would be accepted under the requirements in Table 6 on page 32. If an HB accepts the access and responds, the I/O bus address is first pre-translated according to Table 6 on page 32. If the address

Personal Use Copy - Not for Reproduction

accesses the area from BIM to TPM for PHB0 (the system-memory-alias space), and if the system-memory-alias space is enabled, then the address is pre-translated by changing the address to be in the range of 0 to (16 MB - 1). The resulting address is then used to select the appropriate TCE in the TCE table of an HB. It does this as follows. The most significant 20 bits of the address (for example, AD[31:12], for PCI) is used as an offset into the TCE table for an HB to select the TCE. Thus, the first TCE maps the addresses 0x00000000 to 0x00000FFF of the Memory Space of the I/O bus; the second entry controls translation of addresses 0x00001000 to 0x00001FFF, and so on. The translated real system address is generated as follows. The Real Page Number (RPN) from the TCE replaces the 20 most significant bits of the address from the I/O bus. The least significant 12 bits from the I/O bus address are used as-is for the least significant 12 bits of the new address.

Thus, the HB's TCE table entries have a one-to-one correspondence with the first *n* pages of the Memory Space of the I/O bus. Each HB has its own TCE table starting address, but software may elect to overlay the TCE tables from different HBs. The size of the Memory Space of the I/O bus that can be mapped to System Memory for a particular HB depends on how much System Memory is allocated to the TCE table for that HB and on how much mappable I/O bus Memory Space is unavailable due to I/O devices which are mapped there (that is, the Peripheral memory and Peripheral I/O spaces are unavailable). The size and location of the HB's TCE table is set up and changed by the *set-64-bit-addressing* OF method for the HB. At the time that OF passes control to the operating system, the 64-bit addressing option of the HB will not be enabled and the TCE table will not have any default size or location. The *set-64-bit-addressing* OF method allows the operating system to set the size of the TCE table to zero while enabling the 64-bit-addressing option of the HB. This would only be used in the case where all I/O devices are 64-bit capable and will always put out an address greater than or equal to 4 GB.

Software Implementation Note: All DMA activity for the I/O devices serviced by an HB must be stopped before setting or changing the size of the TCE table for that HB, or changing the table's location.

Hardware and Software Implementation Note: It may be desirable to have the maximum size of contiguous I/O Memory Space possible that gets translated by TCEs without having to first go through the pre-translation for the system-memory-alias space, and which does not have holes due to the Peripheral Memory or Peripheral I/O Space interference. In order to accomplish this, the hardware or OF can place the Peripheral I/O and Peripheral Memory Spaces for the HB at the largest address possible in the lower 4 GB of address space, and make those areas as small as possible. For example, if the Peripheral

Personal Use Copy - Not for Reproduction

I/O and Peripheral Memory Spaces for an HB are in the upper 1 GB of address space, and if the io-hole is disabled, then there will be 3 GB of contiguous I/O bus Memory Space which can be mapped via TCEs.

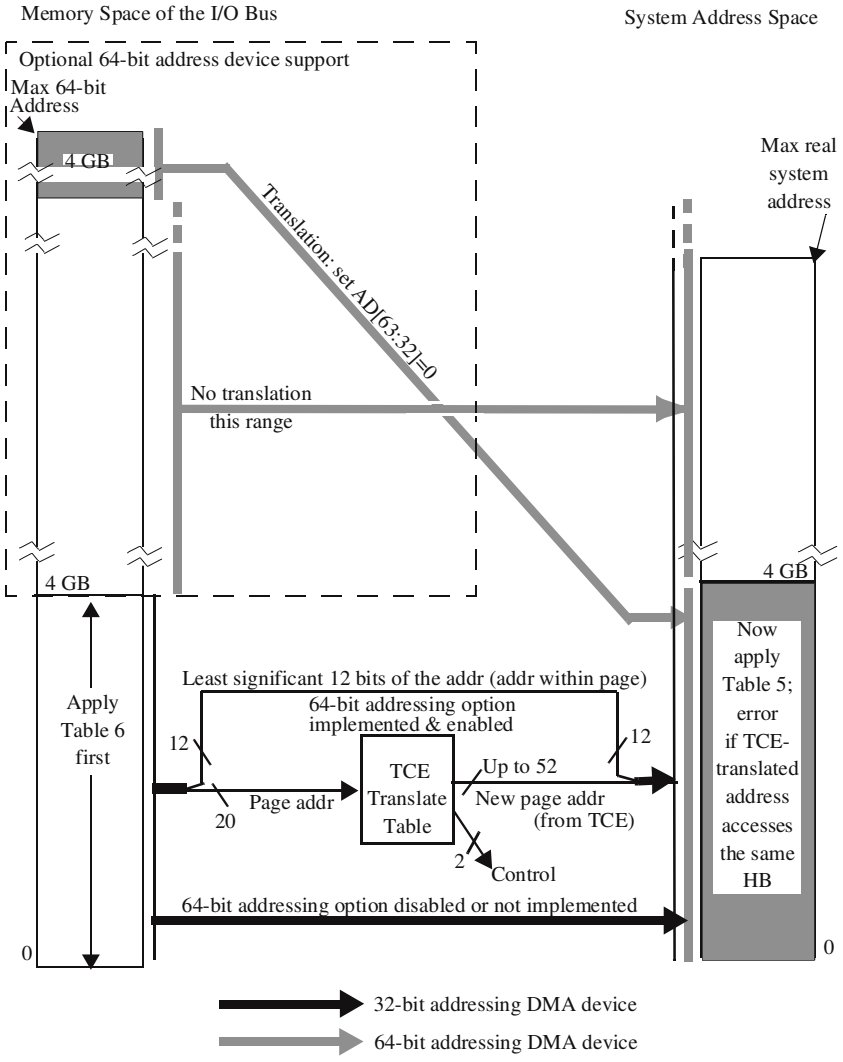


Figure 8. I/O Device DMA Address Translation

Personal Use Copy - Not for Reproduction

Each TCE also contains two control bits. These are used to identify whether that page is mapped to the system address space, and if the page is mapped, whether it is mapped read/write, read only, or write only. See the Table 7 on page 44 for a definition of these control bits.

The HB's TCE table is the analogue of the system translation tables. However, unlike the system translation tables, the dynamic page faulting of memory during an I/O operation is not required (the page fault code point in the TCE control field is used to indicate an error condition and is to be used for error detection).

Table 7. TCE Definition

Bits	Description
0 to 51	RPN: If the page mapping and control field of the TCE indicate anything other than page fault, then these bits contain the Real Page Number (RPN) to which the bus address is mapped in System Memory. In certain HB implementations, all of these bits may not be required, however enough bits must be implemented to match the largest real address in the platform.
52 to 60	Reserved for future use.
61	Reserved for future use (assigned for IBM use).
62 to 63	Page Mapping and Control: These bits define page mapping and read-write authority. They are coded as follows: 00 Page fault (no access) 01 System Memory (read only) 10 System Memory (write only) 11 System Memory (read/write) Code point 0b00 signifies that the page is not mapped. It must be used to indicate a page fault error. Hardware must not change its state based on the value in the remaining bits of a TCE when code point 0b00 is set in this field of the TCE. For accesses to System Memory with an invalid operation (write to a read-only page or read to a write-only page), the HB will generate an error. See Chapter 10, "Error and Event Notification," on page 157 for more information about error handling.

3.3 PC Emulation Option

This section describes an optional set of changes to the system architecture that improves the performance and/or compatibility of a stand-alone emulation environment for PC software. The PC Emulation option consists of minor modifications to the system address map. To support the PC Emulation option, platforms must implement the io-hole and the complete set of additional features described in this section.

Personal Use Copy - Not for Reproduction